SIDLEY AUSTIN BROWN & WOOD LLP

BEIJING
BRUSSELS
CHICAGO
DALLAS
GENEVA
HONG KONG
LONDON

1501 K STREET, N.W.
WASHINGTON, D.C. 20005
TELEPHONE 202 736 8000
FACSIMILE 202 736 8711
www.sidley.com
FOUNDED 1866

LOS ANGELES
NEW YORK
SAN FRANCISCO
SHANGHAI
SINGAPORE
TOKYO
WASHINGTON, D.C.

WRITER'S DIRECT NUMBER
(202) 736-8914

WRITER'S E-MAIL ADDRESS
jkushan@sidley.com

Part of Paper #31

October 22, 2003

**By Hand Delivery**

Stephen G. Kunin
US Department of Commerce
US Patent & Trademark Office
P/DCPEP
Crystal Park 2, 9th Floor
Arlington, VA

Re: WWWTALKQ2 and WWWTALKQ3

Dear Mr. Kunin:

Enclosed please find a compact disk with the above files on it.

Please feel free to contact me if you have any questions.

Sincerely,

Jeffrey P. Kushan

JPK:tnh

Enclosure

SIDLEY AUSTIN BROWN & WOOD LLP IS A DELAWARE LIMITED LIABILITY PARTNERSHIP PRACTICING IN AFFILIATION WITH OTHER SIDLEY AUSTIN BROWN & WOOD PARTNERSHIPS

Paper #31

**October 14, 2003**

Mail Stop: ___ (Citation of Prior Art per 37 CFR 1.501)
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

RE: Submission of Information Pursuant to 35 U.S.C. § 301 and 37 CFR 1.501 in relation to U.S. Patent No. 5,838,906

Dear Sir,

Enclosed, please find prior art publications to be included in the file wrapper of U.S. Patent No. 5,838, 906 ("the '906 patent") pursuant to 35 U.S.C. § 301 and 37 C.F.R. § 1.501.

The '906 patent claims Web browser computer programs and processes. The two references provided herewith are printed publications published more than one year prior to the filing date of the '906 patent. Each is prior art under 35 U.S.C. § 102(b) to the '906 patent. The two printed publications provided herewith were not cited, made of record or considered during the prosecution of the '906 patent. One set of copies is provided for inclusion in the file wrapper of the '906 patent. The second set of copies is provided to permit service by the Office on the patent owner.

One or more claims of the '906 patent are *prima facie* anticipated and/or obvious by the art being cited herein. The '906 patent also has gained significant notoriety in the Internet community because, due to its invalidity, it will unfairly and significantly impact a very wide audience of consumers and other users of the Internet. Examples of press coverage illustrating these concerns are enclosed as Attachment D. The undersigned represent a significant cross-section of the community of developers of Internet-related software who share these concerns, and believe the Director should, on his initiative, commence a reexamination of the '906 patent.

**Existence of a Substantial New Question of Patentability**

The '906 patent generally relates to the ability of a Web browser to handle an "object" in a Web page having a data format not natively supported by the Web browser and thus requiring an external "executable application" to display the object. The '906 patent acknowledges that certain prior art Mosaic browsers allowed users, through clicking on a link, to view and interact with such an object via a "helper application" in a separate window. In such prior art browsers, in response to a user's click on a link, the browser invokes the helper application is invoked to display the object in a separate window. Pursuant to the claims of the '906 patent, in response to the inclusion of an "EMBED text format," or tag, in the document, the browser automatically invokes the helper application to display the object "in-line" in the browser window.

The two enclosed references describe and relate to characteristics of Web browsers for implementing HTML standards. They are dated more than one year prior to the filing date of the '906 patent. They each describe the use of an EMBED tag to automatically invoke an external executable application in order to display and enable interactivity with an object in-line within

the browser window. They each also inherently describe Web browsers including, in particular, the admitted prior art Web browsers of record. As such, each publication describes each claimed element of the inventions defined by at least claims 1 to 3 and 6 to 8 of the '906 patent and as such each publication anticipates these claims of the '906 patent. Alternatively, the newly cited printed publications, when considered in view of the admitted prior art Web browsers of record, render *prima facie* obvious the claimed subject matter of at least claims 1 to 3 and 6 to 8 of the '906 patent. As such, the two enclosed references each raise a substantial new question of patentability regarding the '906 patent.

**Acknowledged Prior Art**

The '906 patent acknowledges that Web browser computer programs were in the prior art. See, e.g., column 2, lines 9 to 12, which provides: "An example of a browser program is the National Center for Supercomputing Application's (NCSA) Mosaic software developed by the University of Illinois at Urbana/Champaign, Ill." More specifically, the inventors of the '906 patent indicate that the subject matter claimed as their invention concerns modifications of certain acknowledged prior art Web browser programs. See, e.g., column 8, lines 9 to 12, of the patent specification, which provides:

> "[t]he source code in Appendix A includes NCSA Mosaic version 2.4 source code along with modifications to the source code to implement the present invention" (emphasis added);

and column 13, lines 43 to 46 which provides:

> "that much of the source code in is [sic] pre-existing NCSA Mosaic code. Only those portions of the source code that relate to the new functionality discussed in this specification should be considered as part of the invention."

The inventors thus acknowledge that the features of Web browsers, at least to the degree reflected in version 2.4 of the NCSA Mosaic Web browser, are prior art to the claimed inventions.

Version 2.4 of the NSCA Mosaic Web browser, like all Web browsers, is a computer program that is implemented on and operated using a computer. The Mosaic program is designed to and preferably runs on a computer connected to the Internet to allow the user to retrieve documents over the Internet and display those documents on the computer. Such documents may contain "an icon, or other indicator, within the text" linked to a particular image file (column 2, lines 64 to 65) that users "may select ... to obtain the full image" (column 3, lines 2 to 3). As the '906 patent admits, when a user selects such an indicator, the Mosaic program "retrieves the corresponding full image ... and displays it by using external software" (column 3, lines 5 to 6) "in a separate window" (column 3, line 17). See generally column 2, line 56 through column 3, line 26 of the '906 patent where the patent describes the capabilities of the Mosaic browser, among others.

**Differences Between the Claimed Invention and the Acknowledged Prior Art**

The differences between the claims and the acknowledged prior art are nominal. Specifically, independent claims 1 and 6 require the computer program/process to process an "EMBED text format," or tag, which is used to automatically display, and enable interaction with, an external object within the browser document window via an external application. The '906 patent asserts that this was an improvement over the prior "helper application" technology employed by prior art browsers such as the Mosaic program in which the browser interprets a user selection of an embedded link to launch an associated external program in a separate window for data that the browser could not process natively. See generally column 3, lines 2 to 20 of the '906 patent.

The patent disclosure and claims specify that the EMBED functionality is expressed in terms of a tag that "specifies the location of ... an object," having "type information associated with it utilized by the browser to identify and locate an executable application," where the tag is parsed by the browser "to automatically invoke said executable application ... in order to display said object and enable interactive processing of said object" in the browser window. See, in particular, Table II of the '906 patent, appearing at column 12, line 54, along with the descriptive text associated with the table appearing at column 13, line 31. These portions of the specification of the '906 patent show that the preferred embodiment of the claimed invention involves use of an EMBED tag having an HREF attribute for specifying the location (e.g., a uniform resource locator, or URL) of an object to be displayed and a TYPE attribute for the MIME type of the object data, which the browser uses to identify, locate and launch an associated application to render that data.

**Prior Art Being Submitted Herewith**

1. David Raggett, HTML+ (Hypertext markup language) (July 23, 1993) (hereinafter "*Raggett I*").

*Raggett I* ("A proposed standard for a light weight presentation independent delivery format for browsing and querying information across the Internet") describes and discloses the functionality of Web browsers that comply with the draft HTML+ specification as of July 23, 1993 (i.e., more than one year before the filing date of the '906 patent). In particular, at page 6, lines 43 to 45, *Raggett I* indicates that such browsers must parse and process "the EMBED tag" contained within a document retrieved over the Internet. *Raggett I* discloses that the EMBED tag includes a TYPE attribute with information concerning the type of the embedded object data. The TYPE attribute, according to *Raggett I*, uses the well-known MIME protocol to enable the browser to identify, locate and invoke an external program to display foreign object data within the document being rendered ("the *type* attribute specifies a registered MIME content type and is used by the browser to identify the appropriate shared library or external filter to use to render the embedded data, e.g., by returning a pixmap"). As is the case with all other HTML tags described in *Raggett I*, the browser performs the related operations for the disclosed EMBED tag automatically upon parsing the tag, without user input.

According to *Raggett I*, "embedding" (page 6, line 40) of an object means displaying the object within the document being rendered. For example, *Raggett I* shows the use of the

EMBED tag to invoke an external program to display an equation or graphic directly in the display of the HTML-based Web page (see, page 6), and also discusses the use of the EMBED tag in combination with the FIG tag to display, for instance, "simple graphs etc. defined in an external format" (page 12, line 30) in the document being rendered and allow for "control of picture alignment and text flow" (page 12, line 17) among other things. See also, generally, page 12, line 13 to page 14, line 6. At page 6, line 47, *Raggett I* further discloses the use of external editor programs that allow for interaction with the displayed object data within the document ("Sophistocated [sic] browsers can link to external editors for updating and revising embedded data"). The '906 patent discloses a comparable TYPE attribute of an EMBED tag (Table II) and use of the MIME protocol for matching the type information to an external program for displaying foreign data within a Web browser window as is described in *Raggett I*.

The above-recited publication was widely disseminated in 1993 by and to, among others, the leaders in the efforts to standardize the Internet, who later became founding participants in the WWW Consortium (or "W3C", the leading standard-setting organization for the Internet). The publication was, has been and continues to be available to all interested persons through the Internet and through other means since on or prior to July 23, 1993.[1] As such, it is a "printed publication" within the meaning of 35 U.S.C. § 102(b). See M.P.E.P. § 2128 (2003).[2] The effective date of the printed publication is the date of its availability; namely, at least as early as July 23, 1993. See M.P.E.P. § 2128.[3] See also, the enclosed declaration from *Raggett I*'s author, David Raggett, which further authenticates the content and date of availability of the publication.

2. Posting of Dave Raggett, dsr@hplb.hpl.hp.com, to www-talk@nxoc01.cern.ch (June 14, 1993) (posting to WWW-Talk public mailing list) (hereinafter "*Raggett II*").

*Raggett II* is an email posting to the WWW-Talk email list (a public, archived and indexed discussion forum) by the author of *Raggett I* (the HTML+ draft specification) that was published on June 14, 1993.[4] It specifically discusses the implementation of the EMBED tag operation disclosed in the draft specification and further notes, in the "p.s.," that the foreign data that is to be rendered in-line by the external editor program need not be contained in the Web document, but may also be external to the document, referenced by a URL. (Compare the '906 patent, e.g., column 13, lines 27 to 28 ("HREF specifies a URL address as discussed above for a data object.").) In addition to repeating the operative description of the EMBED tag operations

---

[1] For example, a dated copy of the document currently can be retrieved from the Cite Seer: Scientific Research Digital Library site via http://citeseer.nj.nec.com/raggett93html.html. Also, dated entries in the WWW-TALK archives related to the referenced provisions of the HTML+ specification, as well as the original posting of the July 23rd HTML+ specification, are still available on-line today at http://ksi.cpsc.ucalgary.ca/archives/WWW-TALK/www-talk-1993q2.messages/467.html and http://ksi.cpsc.ucalgary.ca/archives/WWW-TALK/www-talk-1993q3.messages/282.html.

[2] M.P.E.P. § 2128 provides, in the section entitled "ELECTRONIC PUBLICATIONS AS PRIOR ART: Status as a 'Printed Publication,'" that: "An electronic publication, including an on-line database or Internet publication, is considered to be a 'printed publication' within the meaning of 35 U.S.C. 102(a) and (b) provided the publication was accessible to persons concerned with the art to which the document relates."

[3] M.P.E.P. § 2128 provides, in the section entitled "ELECTRONIC PUBLICATIONS AS PRIOR ART: Date of Availability," that: "Prior art disclosures on the Internet or on an on-line database are considered to be publicly available as of the date the item was publicly posted. If the publication does not include a publication date (or retrieval date), it cannot be relied upon as prior art under 35 U.S.C. 102(a) or (b)."

[4] The complete archives of the WWW-talk email list for the second and third quarters of 1993 are provided on the enclosed CD. The complete archives, or the individual posting, are each printed publications.

from the HTML+ specification, the body of the posted Raggett email provides guidance regarding how to connect a MIME type via an EMBED tag to the appropriate external rendering program ("e.g. via X resources or a config file") and regarding use of external programs and inter-process communications ("separate programs driven via pipes and stdin/stdout or as dynamically linked library modules (Windows DLLs)").

The above-recited publication was widely disseminated and publicly available through the Internet and through other means at least from June 14, 1993, and continues to be available on-line at http://ksi.cpsc.ucalgary.ca/archives/WWW-TALK/www-talk-1993q2.messages/467.html. It is thus a "printed publication" within the meaning of 35 U.S.C. §102(b) because it was a "contribution" to "electronic bulletin boards, message systems, and discussions lists" that were "accessible to the persons concerned with the art to which the document relates" when it was posted to the WWW-Talk list (see, e.g., M.P.E.P. § 707.05(e)).[5] It enjoys prior art effect as from the date of its posting (i.e., June 14, 1993), pursuant to M.P.E.P. § 2128, as noted above.

**Comparison of the Claims to the Acknowledged and Newly Cited Art**

In the context of independent claims 1 and 6, the NCSA Mosaic version 2.4 browser is a "computer program product" (e.g., a Web browser) that is "embodied" in a "computer usable medium" (e.g., installed in a computer or contained on a disk ) for use in a "distributed hypermedia environment" having "at least one client workstation and one network server" (e.g., the Internet). The Mosaic program can run on "said client workstation" to "parse[] a first distributed hypermedia document" (e.g., an HTML document) "received over" the Internet to "identify text formats" (e.g., HTML tags and elements) and "respond[] to predetermined text formats to initiate processing specified by said text formats" in the hypermedia document in order "to display" the document in a browser window on "said client workstation." Furthermore, the Mosaic program can locate "an external object" having "type information associated with it utilized by said browser to identify and to locate an executable application external to" said hypermedia document. The Mosaic program can "invoke" said external application (e.g., an "external editor") "to display" the "external object." As implemented in version 2.4, said invocation and display occurs via another window (as opposed to within the browser window displaying the hypermedia document as required by the claims) when the user selects a hyperlink to the external object (as opposed to "automatically" as required by the claims). Version 2.4 of Mosaic also enables "interactive processing of" (e.g., editing of) the "external object." See, e.g., column 6, lines 32 to 35 of the '906 patent (i.e., prior art browsers permit some degree of interactive processing of the external object).

The only claim limitation not explicitly disclosed, described and implemented in the admitted prior art Mosaic Web browser is the proviso requiring the Web browser to parse an "embed text format" in a hypermedia document to "automatically invoke" an external application "to display" an external object within the browser window displaying the hypermedia

---

[5] For instance, a review of the University of Calgary archive site containing this posting demonstrates that more than 1,000 such postings were made during the three months surrounding the posting of the July 23rd HTML+ Specification (*Raggett I*) by the very people that were developing the Internet at the time. (See <http://ksi.cpsc.ucalgary.ca/archives/WWW-TALK/www-talk-1993q3.index.html>.) Moreover, the HTML+ Specification itself asks that comments be sent "to the WWW discussion group: www-talk@nxoc01.cern.ch." (*Raggett I* at page 1, footnote 1.)

document. *Raggett I* (i.e., the draft HTML+ specification), however, specifically describes just such an HTML "embed" tag for such purposes (i.e., automatically invoking an external program to render interactive objects in-line in an HTML document). This is reflected in the HTML+ specification and in the specification author's contemporaneous email to the WWW-Talk email list, both of which demonstrate that it was well-known in the browser field prior to the filing date of the '906 patent that the foreign data could be contained in a separate file referenced, for example by a URL. Moreover, the ability of a Web browser to retrieve and process data from both local and non-local sources is the inherent design of such browsers. Indeed, one of the first applications of HTML/Web browsers was the rendering, in a single document, of text and image files, where the image files were located in a file external to the file containing the text to be rendered.

An element by element comparison of claim 6[6] to the acknowledged and newly cited prior art is provided below in Table I. It should be noted that, as described in greater detail below, *Raggett I* and *II* each inherently describe each feature of the NCSA Mosaic version 2.4 Web browser, which is acknowledged by the owner of the '906 patent to be prior art.

**Table I**

| Claim 6 | Acknowledged and Newly Cited Prior Art |
|---|---|
| *A computer program product for use in a system having at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment, the computer program product comprising: a computer usable medium having computer readable program code physically embodied therein, said computer program product further comprising: computer readable program code for causing said client workstation to execute a browser application* | Mosaic, see '906 patent at column 1, line 19 to column 3, line 51 (describing Internet, and use and function of browser programs, and noting that Mosaic is "an example of a browser program"). |
| *to parse a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and to respond to predetermined text formats to initiate processes specified by said text formats;* | Mosaic, see '906 patent at column 1, line 19 to column 3, line 51 (same); *Raggett I* at page 3, lines 4 to 38 (discussing "Parsing HTML+ Documents"). |
| *computer readable program code for causing said client workstation to utilize said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server,* | Mosaic, see '906 patent at column 1, line 19 to column 3, line 51 (same). |
| *wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation,* | Mosaic, see '906 patent at column 1, line 19 to column 3, line 51 (same). |
| *wherein said first distributed hypermedia* | Mosaic, see '906 patent at column 1, line 19 to |

[6] Note that claims 1 and 6 are nearly identical but for the type of invention (i.e., claim 1 claims a process, whereas claim 6 is directed to a "computer program product for use in...").

| Table I | |
|---|---|
| **Claim 6** | **Acknowledged and Newly Cited Prior Art** |
| *document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document,* | column 3, line 51 (same, including: "A distributed hypertext or hypermedia document typically has many links within it that specify many different data objects located in computers at different geographical locations connected by a network."); *Raggett II* at pages 1-2 (providing example of embedded text format and stating that: "The browser identifies the format of the embedded date from the "type" attribute, specified as a MIME content type;" and that "you can also put the foreign data in a separate file referenced by a URL"). |
| *wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document* | Mosaic, see '906 patent at column 3, lines 5 to 6 (the Mosaic program "retrieves the corresponding full image … and displays it by using external software") (emphasis added); *Raggett II* at page 1 (providing example of embedded text format and stating that: "The browser identifies the format of the embedded data from the "type" attribute, specified as a MIME content type;" and that "The functions could be implemented as separate programs …") (emphasis added). |
| *and wherein said embed text format is parsed by said browser to automatically invoke said executable application on said client workstation* | Mosaic, see '906 patent at column 1, line 19 to column 3, line 51 (noting that Mosaic is "an example of a browser program" and, as such, parses HTML documents accessed); *Raggett I* at page 3, lines 4 to 38 and page 6, lines 40 to 45 (discussing "Parsing HTML+ Documents" generally, and "the EMBED tag" specifically, as part of the initial processing of every HTML document accessed by a Web browser); *Raggett II* at page 1 (providing example of embedded text format and stating: "The browser identifies the format of the embedded data from the "type" attribute, specified as a MIME content type."). |
| *in order to display said object* | Mosaic, see '906 patent at column 3, lines 5 to 6 (the Mosaic program "retrieves the corresponding full image … and displays it by using external software"). |
| *and enable interactive processing of said object* | Mosaic, see '906 patent at column 6, lines 32 to 35 ("Also, while the present open distributed hypermedia system on the Internet allows users to locate and retrieve data objects it allows users very little, if any, interaction with these data objects."); *Raggett I* at page 6, line 47 ("Sophistocated [sic] browsers can link to external editors for updating and revising embedded data."). |

| Table I | |
|---|---|
| **Claim 6** | **Acknowledged and Newly Cited Prior Art** |
| *within a display area* | Mosaic, see '906 patent at column 3, lines 5 to 6 (the Mosaic program "retrieves the corresponding full image ... and displays it by using external software"). |
| *created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.* | *Raggett I* at page 6, lines 40 to 45 and page 12, line 13 to page 14, line 6 (discussing various options when displaying embedded objects in-line, such as text wrapping around the object) and at page 34, lines 1 to 20 (in section entitled "Notes for Implementers" stating: "It is generally better to avoid displaying the retrieved document in a new window, unless explicitly requested by the user ...."); *Raggett II* at page 1 ("Well both of these will be possible with the HTML+ DTD, by using the capability to embed foreign formats inline in the HTML+ source ...") (emphasis added). |

**The Newly Cited References Anticipate Claims 1, 2, 3, 6, 7 and 8**

As shown above, *Raggett I* and *II* each fully disclose the allegedly novel features of claims 1 and 6; namely, the use of an EMBED tag to automatically invoke an external application to display an external object inline within the same browser window displaying the document containing the EMBED tag. The remaining limitations of claims 1 and 6 are all admitted by the inventors of the '906 patent to be disclosed in prior art Web browsers such as Mosaic. See column 8, lines 9 to 12 and column 13, lines 43 to 46. Those same prior art Web browsers are inherently disclosed and described by *Raggett I* and by *Raggett II*, making each reference fully anticipatory.

*Raggett I* and *II* each refer to Web browsers that are acknowledged to be prior art in the '906 patent (see, e.g., *Raggett I*, page 15, lines 43 to 45). The inherent features and characteristics of such Web browsers, such as Mosaic, include the ability to render HTML-compliant documents. HTML is the predecessor standard to the HTML+ specification that is the basis of the *Raggett I* and *II* disclosures. The set of elements that make up the HTML specification is found in its entirety in, and is added to by, the HTML+ specification. Both HTML and HTML+ are implementations of the Standard Generalized Markup Language (SGML). Consequently, references in *Raggett I* and *II* to prior art Web browsers inherently are described by the disclosure of HTML in *Raggett I* and *II*.

Moreover, those of skill in the browser coding art, upon reading *Raggett I* and *II*, would immediately infer the inclusion of such prior art browsers in the teachings of these two disclosures. See M.P.E.P. § 2144.01 (2003) ("[I]n considering the disclosure of a reference, it is proper to take into account not only specific teachings of the reference but also the inferences which one skilled in the art would reasonably be expected to draw therefrom.") (quoting *In re Preda*, 401 F.2d 825, 826, 159 USPQ 342, 344 (CCPA 1968)). This stems from the fact that

*Raggett I* and *II* each define and describe the functional and other characteristics of computer programs that are HTML+ compliant Web browsers. The discussion in *Raggett I* and *II* concerning new features that prior art browsers should be modified to incorporate *necessarily* includes a full description of the prior art Web browsers themselves. See *Atlas Powder Co. v. Ireco, Inc.*, 190 F.3d 1342, 1346, 51 USPQ2d 1943, 1946 (Fed. Cir. 1999) ("[A] prior art reference may anticipate when the claim limitation or limitations not expressly found in that reference are nonetheless inherent in it. Under the principles of inherency, if the prior art necessarily functions in accordance with, or includes, the claimed limitations, it anticipates.") (internal citations omitted).

Included -- through explicit references and inherently due to the fact that the HTML+ specification builds upon and expands the original HTML specification -- in the disclosure of the HTML+ specification by *Raggett I* and *II* is the original HTML specification See, e.g., *Raggett I* at page 2, line 3 ("HTML+ follows on from an earlier standard - HTML. see [Berners-Lee 93a]."), at page 3, line 40 ("This format is designed to be largely compatible with the earlier format HTML.") and at page 33, lines 1 to 37 (discussing compatibility with HTML, for example, by listing and describing each obsolete tag from HTML and how to map to HTML+). Because the HTML+ specification, like the earlier HTML specification, describes the functionality that Web browsers must possess to be fully compliant with the specification, one of skill in the art would immediately "envisage" both the prior art Web browsers that support HTML and the modified versions of those browsers that comply with the new HTML+ specification. See M.P.E.P. § 2131.02 (in chemical context, stating that a reference may be relied upon for what one of skill in the art would "at once envisage" upon reading the reference).

Particularly when they are considered in light of their inherent disclosures of admitted prior art Web browsers, *Raggett I* and *II* disclose and therefore anticipate each claimed limitation of claims 1 and 6 of the '906 patent. Furthermore, as claims 2, 3, 7 and 8 recite only inherent features present in prior art Web browsers, these claims add no further limitations relative to claims 1 and 6 that would distinguish them from the anticipating disclosures of *Raggett I* and *II*.

**Claims 1, 2, 3, 6, 7 and 8 are Also *Prima Facie* Obvious Over the Prior Art**

As set forth above, claims 1, 2, 3, 6, 7 and 8 are anticipated by *Raggett I* and *II*. In the alternative, these claims are *prima facie* obvious when the acknowledged prior art is taken in view of *Raggett I* and *Raggett II* because these disclosures specifically suggest modifying the prior art to incorporate the differences between the claims and the acknowledged prior art.

**The Level of Ordinary Skill in the Art for Purposes of Obviousness**

The person of ordinary skill in the relevant art to the claimed invention is a software programmer. The '906 patent acknowledges that the act of modifying the Mosaic prior art browser to implement the functionalities described and claimed in the patent was well within the skill of the art. For example, at column 13, lines 51 to 59, the patent states:

> "In general, the flowcharts in this specification illustrate one or more software routines executing in a computer system such as computer system 1 of FIG. 1. The routines may be implemented by any means as is known in the art. For example, any number of

computer programming languages, such as 'C', Pascal, FORTRAN, assembly language, etc., may be used. Further, various programming approaches such as procedural, object oriented or artificial intelligence techniques may be employed."

In addition, at column 16, lines 51 to 53, the patent specifies that:

> "It will, however, be evident that various modifications and changes may be made thereunto without departing from the broader spirit and scope of the invention as set forth in the appended claims. For example, various programming languages and techniques can be used to implement the disclosed invention. ... Many such changes or modifications will be readily apparent to one of ordinary skill in the art."

Thus, based on the admissions within the '906 patent, a software programmer could readily implement the noted functionality into the acknowledged prior art Mosaic Web browser, the source code for which was readily available (also as acknowledged in the '906 patent specification).

**The *Prima Facie* Obviousness of Claims 1 and 6**

The printed publications provided herewith were not considered by the PTO during the original prosecution of the '906 patent. When they are considered in view of the acknowledged prior art (e.g., the version 2.4 Mosaic Web browser) by a person of ordinary skill in the art, they render the claimed invention defined by claims 1 and 6 of the patent *prima facie* obvious.[7]

As noted above, the differences between the claimed invention and the acknowledged prior art Mosaic version 2.4 Web browser are limited to the Web browser parsing an "embed text format" in a hypermedia document (e.g., an HTML document) to "automatically invoke" an external application "to display" an external object within the browser window displaying the hypermedia document. *Raggett I* and *Raggett II* each specifically disclose implementing this functionality in Web browsers.

The two printed publications provided herewith thus provide specific motivation and guidance to a person of ordinary skill to modify the acknowledged prior art NCSA Mosaic version 2.4 browser (and other prior art browsers) to arrive at the claimed invention. Indeed, for a Web browser to be fully complaint with *Raggett I* (the HTML+ specification), which was publicly disseminated more than a year prior to the filing date of the '906 patent, the Web browser must possess the functionality disclosed therein. As such, it is difficult to envision a document that could provide more specific motivation to modify prior art Web browsers to provide the disclosed functionality. Furthermore, as acknowledged and admitted by the inventors of the '906 patent (e.g., column 13, lines 51 to 59 and column 16, lines 51 to 53), the act of modifying the Mosaic prior art browser to implement the features called for by *Raggett I* was well within the abilities of a person having an ordinary level of skill in the relevant art (e.g.,

---

[7] Pursuant to M.P.E.P. §2143 (2003), "[t]o establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations."

software programming). Thus, modification of prior art Web browsers (e.g., NCSA Mosaic version 2.4) by such a person to implement the functionalities described in *Raggett I* or in *Raggett II* would have been *prima facie* obvious to a person of ordinary skill in the art. . Further comparison of the '906 patent specification to *Raggett I* and *Raggett II* confirms this conclusion. As noted above, Table II (column 12, line 54, with descriptive text through column 13, line 31) of the '906 patent shows the preferred embodiment of an EMBED tag with HREF and TYPE attributes which the browser uses to identify, locate and launch associated external applications. The EMBED tag TYPE and HREF attributes, and their descriptions, disclosed in Table II of the '906 patent and the surrounding text are nearly identical to the EMBED tag TYPE attribute disclosed in *Raggett I* (page 6, lines 43 to 46) and to the HREF attribute disclosed elsewhere in *Raggett I* (compare '906 patent at column 13, lines 27 to 28 ("HREF specifies a URL address as discussed above for a data object."), with *Raggett I*, page 13, line 23 (defining HREF as: "A URL specifying the link to traverse when clicked.")). The enclosed publications thus disclose not only the same functionality but precisely the same means of implementing the same functionality in Web browsers (i.e., the same "EMBED" tag is used to initiate the same browser behavior that provides the same results as the claimed subject matter of the '906 patent).

Moreover, the enclosed publications enable, as the '906 patent claims, Web browsers to provide the user with more functionality (e.g., through displaying and/or editing new data formats) without changing the browser code. Compare, '906 patent, column 11, lines 52 to 55, *Raggett I*, page 6, and *Raggett II*, cover page. As noted above, the enclosed publications were promulgated to the WWW community more than a year before the filing of the '906 patent for the purpose of implementing this very same capability in prior art Web browsers.

Attached Exhibit

**Claims 2, 3, 7 and 8 are *Prima Facie* Obvious**

Claims 2 and 7 of the '906 patent add an additional limitation to claims 1 and 6 respectively; namely, that the process or computer program provide for "interactively controlling" the external application "via inter-process communications" between the browser and the external application. The patent specification indicates that "inter-process communications" are a "protocol to exchange information between browser client and application client", and exemplify such communications by referring to the prior art "XEvent interprocess communication protocol" (column 9, line 8 to 10). See also column 16, lines 29 to 32, wherein the '906 patent discusses how "the browser process, Mosaic, communicates with the [external application] process via inter-client communications mechanisms such as provided in the X-Window environment." (Emphasis added.) The added limitations specified in claims 2 and 7 thus refer to characteristics and properties of the acknowledged prior art.

As noted above, claims 1 and 6 are *prima facie* obvious over the acknowledged prior art Mosaic version 2.4 Web browser when taken in view of *Raggett I* and *II*, independently and in combination. The acknowledged prior art, along with *Raggett I* and *II*, also disclose the additional limitation of claims 2 and 7 as noted above. For example, *Raggett I* discloses that "[s]ophistocated [sic] browsers can link to external editors for updating and revising embedded data" (see page 6, line 47). Similarly, *Raggett II* notes that such "separate programs" (e.g., "external editors") can be "driven via pipes and stdin/stdout" (see cover page). An "external editor" is, by definition, a controllable external application, and "pipes and stdin/stdout" is an example of "inter-process communications" for use in transferring data between, among other

programs, a browser and an external application.[8] *Raggett I* and *II,* thus, clearly disclose the additional limitation of claims 2 and 7 and provide specific motivation to one of ordinary skill in the art to modify the NCSA Mosaic version 2.4 Web browser to incorporate the above-noted claimed features. Also as noted above, the '906 patent indicates that a person of ordinary skill in the art has the requisite abilities to implement such features (e.g., column 13, lines 51 to 59 and column 16, lines 51 to 53). Claims 2 and 7, thus, are *prima facie* obvious when the acknowledged prior art NCSA Mosaic version 2.4 Web browser is taken in view of *Raggett I* and *II*, considered individually or collectively.[9]

Claims 3 and 8 add a further limitation calling for "the communications to ... continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched." Similar to the discussion in footnote 9 above, this limitation, however, adds nothing to claims 2 and 7 (or even claims 1 and 6) of the '906 patent. To interactively control an external application, as each of claims 1, 2, 6 and 7 requires, the communications between the browser and the external application must continue after the external application is launched. Claims 3 and 8 thus add no patentable distinction and, for the reasons provided above in relation to claims 1, 2, 6 and 7, are *prima facie* obvious in the light of the acknowledged Mosaic prior art browser in combination with *Raggett I* and *II.*

* * *

04/02/2004 08:22:44 PM

---

[8] "Pipes are IPC (interprocess communication) features of the UNIX, Windows, and OS/2 operating systems." See <http://www.linktionary.com/p/pipes.html> (Tom Sheldon's Linktionary.com, an online networking dictionary).

[9] This is not surprising given that the specification of the '906 patent admits that the additional limitation of claims 2 and 7 is a simple use prior art network capability for its intended purpose (column 9, lines 12 to 13 (X-Windows)). Moreover, at a fundamental level, the '906 patent effectively concedes that this limitation cannot render the otherwise obvious claims 1 and 6 patently distinct. Independent claims 1 and 6 already include a limitation requiring the "external application" to "enable interactive processing" of the external object. In other words, claim 1 and 6 inherently include the "interactively controlling ... via inter-process communications" limitation. After all, to "enable interactive processing" (claims 1 and 6), there must be some type of "inter-process communications" between the browser and an "interactively controll[ed]" external application (claims 2 and 7). The additional limitation of claims 2 and 7, if it can even be called a limitation, is therefore an empty one that merely parrots limitations already included in the underlying independent claims 1 and 6, and thus is certainly as obvious as the underlying independent claims.

In conclusion, the two printed publications provided herewith anticipate at least claims 1, 2, 3 and 6, 7 and 8 of the '906 patent. The acknowledged prior art, when taken in view of the newly cited prior art provided herewith also provide specific motivation and guidance to a person of ordinary skill to modify the NCSA Mosaic version 2.4 browser to arrive at the claimed invention. As such, these disclosures render claims 1, 2, 3, 6, 7 and 8 of the '906 patent *prima facie* obvious to a person of skill in the art.

Very truly yours,

James Brimson
Associate General Counsel
America Online, Inc.

Loren Hillberg
Senior Vice President and General Counsel
Macromedia, Inc.

Andrew Culbert
Associate General Counsel
Microsoft Corporation

In conclusion, the two printed publications provided herewith anticipate at least claims 1, 2, 3 and 6, 7 and 8 of the '906 patent. The acknowledged prior art, when taken in view of the newly cited prior art provided herewith also provide specific motivation and guidance to a person of ordinary skill to modify the NCSA Mosaic version 2.4 browser to arrive at the claimed invention. As such, these disclosures render claims 1, 2, 3, 6, 7 and 8 of the '906 patent *prima facie* obvious to a person of skill in the art.

Very truly yours,

James Bramson
Chief Counsel
America Online, Inc.

Loren Hillberg
Senior Vice President and General Counsel
Macromedia, Inc.

Andrew Culbert
Associate General Counsel
Microsoft Corporation

Exhibit 2 to Declaration of Bramson

Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
Phone 408 536.6000
Fax 408 537.6000

15 October 2003

Commissioner for Patents
Attention: Hon. Steven Kunin
Deputy Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

RE: Potential Director-Ordered Reexamination of U.S. Patent No. 5,838,906 pursuant to 35 U.S.C. §303(a)

Dear Deputy Commissioner Kunin:

As a leading company in the software industry, we are writing to you with regard to U.S. Patent No. 5,838,906, Doyle. We urge the Director of the United States Patent and Trademark Office to exercise his authority pursuant to 35 U.S.C. §303(a), and initiate a Director Ordered Reexamination of it. We have reviewed the Criteria for Initiating a Director Ordered Reexamination, dated August 3, 2000, and, while we agree that such reexamination orders should be rare, we believe the present circumstances regarding the Doyle patent meet the stringent criteria.

In particular, we believe that (a) there is "compelling reason" to order reexamination, and (b) at least one claim in the Doyle patent is *prima facie* unpatentable over patents or printed publications. With regard to critera (b), it has come to our attention that patents or publications have been cited to you under the provisions of 35 U.S.C. §301. It is our further understanding that such art raises a substantial new question of patentability, sufficient to justify a reexamination of said Doyle patent.

By this letter, we would like to focus your attention on the first criteria, in particular, the "compelling reason" requirement. Specifically, we believe that "a significant concern about the patentability of the claimed subject matter has been expressed by a substantial segment of the industry, and that there is substantial media publicity adverse to the patent alleging conspicuous unpatentability of the claims."

The Doyle patent has been the subject of widespread concern within the industry to which it pertains. That community includes, in particular, companies, organizations, and individuals that develop web browsers and technology solutions that work within web browsers. In addition, significant concerns have been expressed within the broader community of owners and users of web sites on the Internet regarding changes that would have to be implemented in web browsers to avoid infringing the Doyle patent. Further still, the negative implications of the Doyle patent have been the subject of significant media publicity. In support of this, we direct your attention to recent news articles

15 October 2003

Commissioner for Patents
Attention: Hon. Steven Kunin
Deputy Commissioner for Patents

appearing in the software trade press that discuss the widespread impact of changes to the Internet.

We would note that the Doyle patent is the subject of litigation in the Northern District of Illinois, brought by the assignee, Eolas, Inc. against Microsoft Corp., and that Microsoft has been found to have infringed the current claims. Furthermore, Microsoft has recently announced that they will make changes to their browser to deal with this alleged infringement, and that such changes will affect an enormous segment of the Internet-using community.

Accordingly, we believe that the rare circumstances justifying a Director-ordered Reexamination of the Doyle patent have been met. As it is our understanding that it is your authority to review potential Director-Ordered Reexaminations on behalf of the Director, and make recommendations to him with regard to ordering them, we respectfully request that you favorably consider such a request and recommend to the Director that he order the reexamination of U.S. Patent No. 5,838,906.

Respectfully submitted,

MeMe Jacobs Rasmussen
Associate General Counsel
Adobe Systems Incorporated

10/20/2003 14:44:23 [0000]

Approved For Release 2005/12/14 : CIA-RDP85-00988R000100010001-5

Attachment A

David Raggett, *HTML+ (Hypertext markup language)* (July 23, 1993)

("*Raggett I*")

Request for Comments: XXXX

David Raggett (Hewlett Packard)
23rd July 1993

# HTML+ (Hypertext markup language)

A proposed standard for a light weight presentation independent delivery format for browsing and querying information across the Internet

## Status of this Memo

This document is a proposal for an Internet Draft, and specifies the HTML+ wide-area hypertext document format, with a view to requesting discussion[1] and suggestions for improvements. Distribution of this memo is unlimited.

## Abstract

HTML+ is a simple SGML based format for wide-area hypertext documents, for use within the World Wide Web. Unlike desktop publishing formats, HTML+ captures the logical intent of authors. This simplifies the task of writing documents, and permits them to be effectively rendered on a wide range of display types as well as the printed page.

HTML+ represents a substantial improvement over the existing format: HTML, offering nested lists, figures, embedded data in foreign formats for equations etc, tables with support for titles and column headings, change bars, entry forms for querying and updating information sources and for use as questionaires for mailing. This document specifies the HTML+ format with guidelines on how it should be rendered by browsers.

## Introduction

The World Wide Web is a wide area client-server architecture for retrieving hypermedia documents across the Internet. It also supports a means for searching remote information sources, for example bibliographies, phone directories and instruction manuals. There are three main ingredients:

a) Universal naming scheme for documents. The universal resource location syntax specifies documents in terms of the protocol to be used to retrieve them, their Internet host and path name. A format for location independent lifetime identifiers is currently being defined by working groups of the IETF. A network protocol will allow universal resource numbers (URNs) to be resolved to the URL for the nearest available copy.

b) Use of available protocols for retrieving documents over the network, including FTP, NNTP, WAIS, Gopher, and HTTP. The latter is designed specifically for use with the World Wide Web, and combines efficiency with an ability to flexibly exchange information between clients and servers.

c) A document format supporting hypertext links based on URLs and URNs which can specify documents anywhere in the Internet. HTML+ is designed for rendering on a wide variety of different display types and platforms.

Information browsers can display information in a wide variety of formats, e.g. plain text, rich text in the HTML+ format, images in the GIF and JPEG formats, MPEG movies, and MIME documents. The hypertext format has a special significance as it allows users to navigate from one document to the next at the click of a button. It provides the basis for menus, cross references, either within a document or to other documents,

---

[1]Please mail comments to the author `dsr@hplb.hpl.hp.com`, or to the WWW discussion group: `www-talk@nxoc01.cern.ch`

perhaps on the other side of the world. It also provides a means of building larger scale collections of documents that act as journals, books or encyclopedias. The format is also intended to act as a building block for creating wide area groupware applications.

HTML+ follows on from an earlier standard - HTML, see [Berners-Lee 93a], which has been widely used as the basis for hypertext documents in the World Wide Web. The new format grew out of experience with HTML, culminating in the desire to add new features, e.g. inline images, tables, and form fields for greater flexibility in querying remote information sources. This document specifies the HTML+ format and suggests ways in which browsers can choose to render it on a variety of different display types.

# 2. HTML+ and SGML

HTML+ itself is based on the Standard General Markup Language (SGML), an international standard for document markup that is becoming increasingly important. The term markup derives from the way proof-readers have traditionally pencilled in marks that indicate how the document should be revised.

SGML grew out of a decade of work addressing the need for capturing the logical elements of documents as opposed to the processing functions to be performed on those elements. SGML is essentially an extensible document description language, based on a notation for embedding tags into the body of a document's text. It is defined by the international standard ISO 8879. The markup structure permitted for each class of documents is defined by an SGML Document Type Definition, usually abbreviated to DTD.

Working groups in ISO have recently produced a range of SGML DTDs for documents, e.g. ISO 12083 defines DTDs for books and ISO 10744, which defines the HyTime standard for hypermedia/time-based documents. These standards are large and complex, and perhaps best suited as interchange standards that facilitate conversion between proprietary document formats. By contrast, HTML+ provides a lightweight delivery format that can be rendered by relatively simple browsers, and which has grown out of two years practical experience with wide-area hypertext information systems in the Internet community.

## HTML+ and HyTime

The HyTime standard provides a rich range of architectural forms, but is not aimed at run-time efficiency. Suggestions have been made as to how the HTML DTD could be adapted to comply with HyTime's clink architectural form [Kimber 93]. This would necessitate documents declaring links as external entities and the use of local names in link definitions, but in the absence of any immediate benefit, there has been little enthusiasm for this within the World Wide Web community. Instead, it is believed that a straightforward filter program should be used to map HTML and HTML+ documents into a format which is strictly compliant with HyTime, when this becomes appropriate.

## A simple example of HTML+

The following is a simple example of an HTML+ document,which illustrates the basic ideas involved in SGML.

```
<title>A Simple HTML+ Document</title>

<h1 id="a1">This is a level one header</h1>

<p> This is some normal text which will wrap at the window margin. You
can emphasise <em>parts of the text</em> if you wish. </p>

<p> This is a new paragraph. Notice that unlike title and header tags,
the matching end tag is optional.
```

The text of the document includes tags which are enclosed in <angle brackets>. Many tags have matching end tags for which the tag name is preceded by the "/" character. The tags are used to markup the document's logical elements, for example, the title, headers and paragraphs. Tags may also be accompanied by parameters, e.g. the "id" attribute in the header tag, which is used to define potential destinations for hypertext jumps.

Unlike most document formats, SGML leaves out the processing instructions that determine the precise appearence of the document, for example the font name and point size, the margins, tab settings and how much white space to leave before and after different elements. The rendering software makes these choices for itself (perhaps guided by user preferences), and so can avoid problems with different page sizes or missing fonts.

Logical markup also preserves essential distinctions that are often lost by lower level procedural formats, making it easier to carry out operations like indexing, and conversion into other document formats.

Practical experience has shown that people often make mistakes when they have to type in the markup for themselves. As a result, most browsers are tolerant of bad markup. This problem is being minimised by keeping the format as simple as possible and encouraging the development of WYSIWYG editors.

# The HTML+ Document Format

The following sections go through the various features of the format with suggestions as to how browsers should render them. The DTD for HTML+ is given in Appendix I.

## Parsing HTML+ Documents

By default, HTML+ documents are made up of 8-bit characters in the ISO 8859 Latin-1 character set. In future, 16 bit character sets may be used to cover a wider range of languages. The HTTP network protocol uses the MIME standard (RFC 1341) to specify the document type and the character set. It is assumed that the chosen character set includes the printable 7 bit US ASCII characters as a subset.

The DTD specifies the syntax of the document structure, in particular, which tags are permitted in any given context. Certain tags are only permitted at the start of the document. Tags and attribute names are case insensitive, thus <TITLE> is equivalent to <title>. Minimisation is forbidden to avoid problems with parsing unknown tags.

In general, SGML entity definitions are used to represent characters which would otherwise be confused with markup elements:

| | | |
|---|---|---|
| & | is represented by | `&` |
| < | is represented by | `<` |
| > | is represented by | `>` |

Such entity definitions should be used in all places except within attribute values for tags (tag names and attribute names cannot contain these particular characters). Entity definitions can also be used for special characters, e.g. "é" for a small e with an accute accent. The full list is given in Appendix II. Additional entities may be defined within documents using the SGML entity declaration tag !ENTITY, e.g.

```
<!ENTITY sgml "Standardised General Markup Language">
```

The browser will then insert the full form whenever it comes across "&sgml;".

Repeated white space characters such as space, tab, carriage return, line feed and form feed are ignored except within preformatted text, i.e. it doesn't matter which white space characters you use or how many of them you put between words, or before or after markup elements, the effect is the same as a single space character.

It is strongly recommended that HTML+ documents start with the following external identifier, indicating that the document conforms to the HTML+ DTD. This will ensure that other SGML parsers can process HTML+ documents, without needing to include the DTD with each document.

```
<!DOCTYPE htmlplus PUBLIC "-//Internet/RFC xxxx//EN">
```

HTML+ departs slightly from pure presentation independence by allowing authors to specify rendering hints, e.g. to use a bold font for a given type of emphasis. This step was taken to give authors greater control over the final appearence, and is based upon practical experience with the earlier HTML format. In addition, attribute values are used to distinguish different subcategories of markup, rather than adding extra tags. New logical categories of emphasis etc. can be added in future without needing to change existing browsers. These decisions have made it practical to restrict HTML+ to a very small set of tags.

## Backwards Compatibility with HTML

The format is designed to be largely compatible with the earlier format HTML, and it is recommended that HTML+ browsers continue support for the few tags which have been dropped. This will avoid problems for the large numbers of HTML documents without the DOCTYPE declaration. Suggestions on how to map HTML elements to HTML+ are given in Appendix III.

# Normal Text

This is generally shown with a serif font and wraps on the right window margin. It can include:

- Entity references, e.g. ">" and "é"
- Significant Line breaks (the BR tag)
- Hypertext links - the A tag
- Inlined graphics or icons - the IMG tag
- Various styles of logical emphasis - the EM tag
- Embedded data in an external format, e.g. TeX equations - the EMBED tag
- Input fields for forms - the INPUT tag

## Line breaks and <BR>

Line breaks have a semantic significance in some contexts, e.g. the lines of a poem or a postal address. This tag causes the renderer to start a new line at the current left margin setting. There is no corresponding end tag. The BR tag is *empty*, that is to say, it doesn't act as a container around other text or markup.

## Hypertext Links

When the user clicks on a hypertext link in the document, the current document is replaced by the one referenced by the link. Links can be made to a wide range of document types, based on the URL[2] and URN[3] notations. Some document types permit links to be made to specific sections within a document[4]. The syntax for links within the same document or to documents in the same directory is particularly simple:

```
Links are defined with the <a href="#z1">A tag</a>. HTML+ supports a
number of <a href="links.html">different link types</a>.
```

In a browser this might look like:

Links are defined with the *A tag*. HTML+ supports a number of *different link types*.

The first link is to an anchor named "z1" in the current document. The second is to a file named "links.html" in the same directory as the current document. The caption for the link is the text between the start and end tags. The value for the HREF attribute defines the destination point, and can be abbreviated in certain cases. If practical, word the caption in such a way that continues to make sense when the document is printed out. The link should be shown in a clearly recognisable way, e.g. as a raised button, or with underlined text in a particular color. For displays without pointing devices, it is suggested that a reference number is given in square brackets, which can then be typed by the user.

A more general discussion of hypertext links and their treatment in HTML+ is presented in a later section.

## Inlined Graphics or Icons

These are treated like characters and inserted as part of the text, e.g.

```
This line has a egyptian hieroglyph at the end of the
line. <img src="ankh.tiff">
```

The URL notation is used to name the source of the graphics data. The *align* attribute can be used to control the vertical position of the image relative to the current text line in which the IMG element is placed. Use a value of "top", "middle" or "bottom" to align the top, middle or bottom of the image with the current text line.

---

[2]The notation for universal resource locators is defined in [Berners-Lee 93b].

[3]The notation for universal resource numbers and the protocol for resolving them to the nearest available copy is currently under study by the IETF URN working group.

[4]At the time this document was written, such links were restricted to named anchors within HTML and HTML+ documents

The *seethru* attribute allows authors to include a chromakey, i.e. a colour that designates portions of the image to be left unpainted so that the background shows through. The format for this attribute's value is dependent on the type of graphics data, and has yet to be defined.

Note that you can create simple iconic buttons, e.g.

```
<a href="bigpic.gif"><img src="littlepic.tiff"></a>
```

If the user clicks anywhere on the image, this will cause the browser to retrieve its bigger version. This approach allows users to preview images which may take significant time to download. Note that there is little additional penalty for displaying the same image at multiple points in the document. The *ismap* attribute is provided for backwards compatibility with HTML. When present the browser will send all mouse clicks and drags on the image, to the server.This mechanism is explained in more detail for the FIG tag.

Sophistocated HTML+ editors should allow authors to modify images using an external editor. Larger images should be specified with the FIG tag.which provides support for flowing text around figures, along with captions, overlays and active areas.

## Various Styles of Emphasis[5]

This allows you to emphasise a portion of the text. The simplest approach is:

```
<em>default emphasis, usually shown in an italic font</em>
```

The logical role of emphasis denotes the semantic significance, e.g. a citation, or text to be input by a user for a computer program.The physical style of emphasis controls its appearence. Note that EM elements can include inlined graphics.

### Logical Role of Emphasis

It is strongly recommended that the logical role of the emphasis is given with the *role* attribute, e.g.

```
<em role="cite">a citation</em>
```

Providing a logical role allows browsers to apply differing rendering styles according to the role, but more importantly, it allows indexes to be constructed automatically, e.g. the list of bibliographic references in a technical report. These can be used for searching through collections of documents according to semantic keys giving better focussed searches compared with full text indexes .

The list of recommended roles are as follows: *(this can be given in upper or lower case)*

For references to other works:

| | |
|---|---|
| cite | a reference to a related work |
| pub | a publication containing a referenced work |
| author | an author of a referenced work |
| editor | an editor of a referenced work |
| title | the title of a referenced work |
| credits | e.g. the rights owner of a photograph |
| copyright | the holder of the copyright |
| isbn | for ISBN numbers |
| acronym | for acronyms like "NATO" and "US" |
| abbrev | for abbreviations |

For annotations:

| | |
|---|---|
| footnote | shown as footnote or pop-up |
| margin | shown as margin note or pop-up |

For computer instruction manuals:

| | |
|---|---|
| dfn | defining instance of a term |

[5]The name EM was chosen in preference to EMPH because it allows existing HTML browsers to show all HTML+ emphasis in italics. It also allows HTML+ browsers to correctly process the common case for emphasis in HTML documents.

| | |
|---|---|
| kbd | something a user would have to type |
| cmd | command name, e.g. "chmod" |
| arg | command arguments, e.g. "-I" |
| var | named place holder, e.g. "filename" |
| ins | an instanced of a named printer, directory or file etc. |
| opt | an option of some kind |
| code | an example of code (shown with a fixed pitch font) |
| samp | a sequence of literal characters |

On dumb terminals annotations should be shown in round brackets. Margin notes should be right aligned, and may include graphics via the IMG tag. The set of recommended roles will be kept by the HTML+ registration authority.

### Physical Styles

The appearence can be modified by adding optional rendering hints from the list:

```
<em b>              bold text
<em i>              italic text
<em u>              underlined text
<em sup>            superscript text
<em sub>            subscript text
<em tt>             type writer font (courier)
<em hv>             sans serif font (helvetica)
<em tr>             serif font (times roman)
```

These hints can be combined, e.g.

```
<em b i> for bold italic text </em>
```

Note that these are only hints and may be ignored by browsers. Indeed, arbitrary combinations will present difficulties for most browsers. If the display is limited to a single font, colour or underlining can be used, but should be clearly differentiated from hypertext links and headers. Dumb terminals can use email conventions, e.g. switching to all capitals, or delimiting with the * or _ characters. Subscript and superscript text should be shown in a smaller point size, vertically offset as appropriate.

Browsers may choose to simplify or ignore hints, but should aim to do so in a consistent manner. At the simplest level, browsers can ignore the attributes and render all emphasis in the same style.

### Nested Emphasis

Emphasis can be nested as in:

```
<em b>bold text, and <em i>bold italic text</em></em>
```

Nested emphasis is better suited for grouping logical roles together, for instance, you could use EM to separately tag author, title, and publication, and then wrap these up as a citation. Without this, indexing programs will have difficulty in grouping markup into the correct references.

## Horizontal Rule

The <HR> tag may be used to draw a horizontal rule to separate text sections. It can be rendered as a simple line across the middle section of the window/paper or similar decoration.

## Embedded data in an external format

The EMBED tag provides a simple form of object level embedding. This is very convenient for mathematical equations and simple drawings. It allows authors to continue to use familiar standards, such as *TeX* and *eqn*. Images and complex drawings are better specified using the FIG or IMG elements. The *type* attribute specifies a registered MIME content type and is used by the browser to identify the appropriate shared library or external filter to use to render the embedded data, e.g. by returning a pixmap. It should be possible to add support for new formats without having to change the browser's code, e.g. through using a common calling mechanism and name binding scheme. Sophistocated browsers can link to external editors for creating or revising embedded data. Arbitrary 8-bit data is allowed, but &, < and > must be replaced by their SGML entity definitions. For example <embed type="application/eqn"> 2 pi int sin (omega t)dt </embed> gives

$$2\pi \int \sin(\omega t) dt$$

## Input Fields for Forms

Input fields can be arranged with considerable freedom, as part of normal paragraphs, preformatted text, lists or tables. Examples of how to do this are given later on in the section describing the FORM tag. The INPUT tag has the following attributes:

**name** Used to name this input field, e.g. `name="phone number"` *(required attribute)*.

**type** Defines the type of data the field accepts (the type name is insensitive to upper/lower case). If missing, the field is assumed to be a free text field.

**size** Specifies the size/precision of the input field according to its type, see below *(optional)*.

**value** The initial value for the field, or the value when checked for checkboxes and radio buttons *(optional, except for radio buttons)*.

**checked** When present, this attribute indicates that a checkbox or radio button is selected.

**disabled** When present, this attribute indicates that this field is temporarily disabled. Browsers should show this by greying out or via a similar visual clue. Users are unable to set the focus to disabled fields, or change their values.

**error** When present, this attribute indicates that the current value for this field is in error in some way, e.g. because it violates some consistency constraints. Browsers should indicate this by a change to the shape and colour (red) of the field's border. This should be accompanied by an error message and a beep.

The following types of field should be supported: *(in either upper or lower case)*

text Single or multi-line text entry fields. Use the *size* attribute to specify the width and height in characters, e.g. size="24" or size="32x4".

url For fields which expect document references as URL or URNs.

int For entering integer numbers, the maximum number of digits may be given with the *size* attribute, e.g. size=3 for a 3 digit number[6].

float For fields restricted to floating point numbers.

date Restricted to a recognised date format.

checkbox Use these for simple boolean attributes, or for attributes which can take multiple values at the same time from some set of alternatives (for fields with the same *name*).

radio Use these for attributes which can take a single value from a set of alternatives (groups input fields with the same *name*).

For the purposes of sending the contents of a form to a server, as part of a query, the input fields are mapped to a list of properties. In most cases the *name* and current *value* are used to define a property/value pair for each field. Radio buttons and check boxes are left out of the list if they are unselected. This ensures that only the selected radio button yields a property/value pair. By missing out the *value* attribute for check boxes, these fields will map to a simple (value-less) property. The representation of property lists is defined as part of the HTTP protocol.

Browsers can choose to notify the server whenever a field is changed (i.e. when a field looses the focus and its contents have changed) or wait until the form is completed. This choice will depend on network latency.

Drop down or "combo" style selection lists may be added in a future revision to this standard.

---

[6]Perhaps the syntax should permit integer ranges, e.g. size="1 to 6", in which case a more appropriate name for the attribute than *size* would be desirable.

## Headers and Titles

The title tag is generally used to define the window banner when viewing a particular document, e.g.

```
<title>Reference Guide to HTML+</title>
```

This element should appear at the start of the document. There are six levels of headers, H1 to H6, with H1 the most important, and H6 the least. A common convention is to begin the body of the document with a level one header. e.g.

```
<h1>Introduction to HTML+</h1>
```

Header names should be appropriate to the following section of the document, while the document title should cover the document as a whole. There are no restrictions on the sequence of headers, e.g. you could use a level three header following a level one header. Browsers should render headers with a line break before and after the header text. A common convention for headers is to use a sans serif font, e.g. Helvetica, with a smaller point sizes for less significant headers, and a serif font, e.g. Times Roman, for normal text.

Headers can include an identifier, unique to the current document, for use as destinations of hypertext links, e.g.

```
<h1 id="intro">Introduction to HTML+</h1>
```

This allows authors to make links to particular sections of documents. It is a good idea to use something obvious when creating an identifier, to help jog your memory at a later date. WYSIWYG editors may automatically generate the identifiers. In this case, they should also provide a point and click mechanism for defining links, so that authors don't need to deal explicitly with the identifiers.

The attribute "margin" when present acts as a hint to the browser to insert the header into the margin and causes the following text to be vertically aligned with the start of the margin header. By convention, margin headers are left justified, e.g.

```
<h4 margin> Deleting the Curve </h4>
The Delete command allows you to delete any selected symbol or text
block.
```

Note that headers don't act as containers for the subsequent text. You can group the header and text with the GROUP tag, see later for details.

## Indexing

A good index plays an important role in helping you find your way to the material you need. It allows you to type in one or more keywords to see a list of matching topics. Alternatively you can browse through the index and take advantage of serendipity. This also allows you to gain a feeling for the limits of what is covered. The two approaches can be combined, when the characters typed act dynamically to control the viewing position within the index.

Typically each keyword entry in the index is associated with one or more topics. This notion of guiding the user is absent from full text indexes like WAIS, where users are given very little help in choosing the keywords to search on. Generating a conventional index for a document is a skilled task, and HTML+ allows authors to include directives for creating an index. These directives can be included with document titles, headers and emphasis etc. using the *index* attribute. This allows each such element to be included in one or more entries in the index, under primary or secondary keys, e.g.

```
<h3 id="z23" index="Radiation damage/shielding from as difficult">Radiation shielding</h3>
```

This resulting index looks like:[7]

Radiation damage
  classical target theory
  dominance of
  in molecular mills
  shielding from as difficult
  simple lifetime model
  track-structure lifetime model
Radicals
  *and so on.*

Where each entry is a hypertext link to the associated anchor. The *index* attribute can specify multiple entries, each separated with the ";" character. The optional secondary key (*shielding from as difficult*) is introduced by the "/" character. Secondary keys are useful when the primary key occurs more than once. To allow for future extension, primary keys should not start with the "#" character. This prefix is being reserved to designate indirect index entries. Use "\/", "\;", "\#" and "\\" to escape "/", ";", "#" and "\" respectively.

# Paragraphs and Preformatted Text

HTML+ includes support for paragraphs and preformattted or verbatim text.

## Defining Paragraphs with <P>

The P tag is used to define paragraphs. Unlike many other tags, the end tag is optional. Note, that unlike HTML,the tag acts as a container for the text of the paragraph. This allows paragraphs to act as hypertext anchors. The end of the paragraph is implied by finding markup elements which are not permitted as part of a paragraph.

The following attributes may be used:

| | |
|---|---|
| **id** | An identifier, unique to this document, which can be used as a destination in a hypertext link. |
| **role** | The role of the paragraph, see the following list for supported types. |
| **align** | A rendering hint to the browser to justify lines. The supported values should be: `align="left"`, `align="center"`, `align="right"` and `align="flush"`. This is useful for single line paragraphs or when the lines are made explicit with the <BR> tag. |
| **indent** | When present, this hint suggests that the left and right margins are indented by an amount dependent on the browser, e.g. about 4 character widths. |

The *role* attribute is used to indicate the logical role of the paragraph, e.g. a stanza in a poem or a cautionary note in a computer manual. Browsers may apply particular rendering styles to certain roles. The role name is case insensitive.

The following roles are recommended: *(in upper or lower case)*

| | |
|---|---|
| quote | A paragraph quoted directly from some other work. Browsers could indent the paragraph and maybe use a different font. |
| byline | Information about the author of the document, e.g contact details. This could be displayed in a different font, and perhaps right aligned. |
| note | Advisory note in an instruction manual. The browser could display a hand icon in the margin. |
| caution | Cautionary note. The browser could display an warning road sign in the margin. |
| error | A note describing error conditions. The browser could indicate the importance of the note by displaying a stop sign in the margin. |

[7]Taken from K. Eric Drexlers's "Nanosystems, Molecular Machinery, Manufacturing and Computation".

An example of a paragraph element:

```
<p role="note"> If you accidentally delete a symbol other than the red
circle, immediately press ALT+BKSP to choose the undo command, and
then select the red circle and delete it again.
```

Paragraphs can be rendered by indenting the first line, or by leaving a vertical gap, for example, half the current line spacing. When using the latter style, browsers should take care to avoid inserting this vertical gap when the paragraph element immediately follows a header. This rule ensures that authors can tag paragraphs directly following a header without causing unwanted extra space to appear before the start of the text.

### Preformatted Text with <PRE>

This is generally shown in a fixed pitch font and preserves the original spaces and line breaks. The horizontal tab character is deprecated, but should be interpreted as a move to the next tab setting, at every eighth column. Preformatted text is useful for including plain ASCII text, e.g. program listings and email messages. A number of tags can be included within preformatted text, e.g. hypertext links using the A tag, emphasis, inline images and input fields. The following optional attributes can be used:

| | |
|---|---|
| **id** | An identifier, unique to this document, which can be used as a destination in a hypertext link. Note that the paragraph tag acts as a container for the paragraph. |
| **role** | The role of the element. |
| **tr** | Use a proportional serif font, e.g. Times Roman. |
| **hv** | Use a proportional sans serif font, e.g. Helvetica. |
| **width** | This gives the maximum number of characters which will occur on a line. The default value is assumed to be 80. Browsers recognising this attribute should optimally handle widths of 40, 80 and 132, with other widths being rounded up. |

Preformatted text started off in HTML with a simple mechanism for showing computer output, for which the spaces and line breaks were significant in determining the layout. The desire to render Unix manual pages as hypertext forced a rethink. The new version supported character emphasis and hypertext buttons for cross references. HTML+ adds the capability to use variable pitch fonts, and to set up tab stops, e.g.

```
<tab at=40 align=right>
```

The *at* attribute specifies the position of the tab stop, measured from the left margin in terms of the width of the character M for the current font. The *align* attribute is one of "left", "center", "right" or "decimal", defaulting to left alignment. For greater control of layout, authors should exploit the FIG or EMBED tags to use external formats, for example encapsulated Postscript. Unfortunately these formats don't as yet support hypertext links.

## Ordered, Unordered and Definition Lists

There are three kinds of lists: ordered or numbered lists, unordered lists and definition lists. Ordered and unordered lists can be nested arbitrarily, and browsers should progressively inset the left margin for each level of nesting.

### Ordered Lists with <OL>

The list items are automatically numbered, e.g.

```
<OL>
  <LI>Wake up
  <LI>Get dressed
  <LI>Have breakfast
  <LI>Drive to work
</OL>
```

Is displayed as:

1) Wake up
2) Get dressed
3) Have breakfast
4) Drive to work

The *compact* attribute when present, e.g. <ol compact>, has the effect of reducing interitem spacing. Authors can also make both the OL tag and the LI tag potential destinations for hypertext links with the *id* attribute. List item text can include normal text and paragraph elements, but not headers.

## Unordered Lists with <UL>

These are bulleted lists, e.g.

```
<UL>
  <LI>Wake up
  <LI>Get dressed
  <LI>Have breakfast
  <LI>Drive to work
</UL>
```

Is displayed as a bulleted list:

- Wakeup
- Get Dressed
- Have breakfast
- Drive to work

The *compact* attribute when present, e.g. <ul compact>, has the effect of suppressing bullets and reducing interitem spacing. Multicolumn lists can be requested with the *narrow* attribute, e.g. <ul narrow>. This causes the browser to try to lay out the list as a number of columns, depending on the window width. This attribute should only be used when all the items are less than 20 characters long. Authors can also make both the UL tag and the LI tag potential destinations for hypertext links with the *id* attribute. List item text can include normal text and paragraph elements, but not headers. For nested unordered lists, browsers may use different bullet symbols for different levels, in addition to progressively insetting the left margin. The *src* attribute on the LI tag can be used to specify an icon for use in place of the standard bullet symbols.

## Definition Lists with <DL>

These consists of pairs of terms <DT> and definitions <DD>. The following example is part of a french dictionary:

```
<DL>
  <DT>endetter
  <DD>Engager dans des dettes

  <DT>endeuiller
  <DD>Plonger dans le deuil, remplir de tristesse

  <DT>endiablé, ée
  <DD>D'une vivacité extrême
</DL>
```

Is commonly displayed as:

| | |
|---|---|
| **endetter** | Engager dans des dettes |
| **endeuiller** | Plonger dans le deuil,<br>remplir de tristesse |
| **endiablé, ée** | D'une vivacité extrême |

With the *compact* attribute, e.g. <dl compact>, this could be altered to:

**endetter** Engager dans des dettes

**endeuiller** Plonger dans le deuil, remplir de tristesse

**endiablé, ée** D'une vivacité extrême

In this style, the term and definition appear in the same paragraph, with the term text emphasised in a bold font. The definition text follows on, and wraps to a left margin a little further inset than the term text. This style is common place in dictionaries.

Term text following the <DT> is restricted to normal text. The definition text after the <DD> tag can additionally include paragraph elements and ordered/unordered lists. Headers are not allowed in either case. Authors can make the DL, DT and DD tags potential destinations for hypertext links with the *id* attribute.

Authors are reminded to check that DT and DD are paired up. Common misunderstandings lead to people repeating DD tags to separate paragraphs (use <P> instead), or leaving out the DT tag altogether to indent text (use <p indent> or <group indent>). The ability of browsers to cope with bad markup seems to encourage such problems, which will hopefull fade away as wysiwyg editors become commonplace

# Figures

Figures provide great flexibility, and can be used to show images, graphics or other information specified in an external format:

- linked or embedded definitions
- control of picture alignment and text flow
- Figure description for when the image can't be shown
- caption placement
- scaled or pixel-based coordinates
- hypertext links with active areas
- text and image overlays

The following simple example will set the scene for the description of the various features:

```
<fig align="right" src="map.gif"> How to get to my house </fig>
```

Here, the image is defined by a link to an external document. The caption "How to get to my house" will appear at the bottom of the image. The *align* attribute directs the browser to display the figure at the right of the widow, and to flow subsequent text around the left of the image.

## Using embedded graphics data

Instead of the *src* attribute, you can include an EMBED element immediately following the <fig> tag. This is useful for simple graphs etc. defined in an external format.

## Figure Description

The FIGD tag allows you to give a textual description which can be shown when the figure itself can't be shown, e.g. for browsers working on dumb terminals, e.g.

```
<FIGD> This is an aerial photograph of central London, showing
Buckingham Palace and the Houses of Parliament. On the left you can see
Hyde Park and in front the Albert Hall and the Natural History
Museum.</FIGD>
```

## Alignment and Text Flow

The *align* attribute controls the horizontal position of the figure: "left", "right", or "center". The default is "left". Browsers may flow text when there is sufficient room, unless the figure is center aligned or the *noflow* attribute is present.

## Caption Placement

The *cap* attribute allows you to ask the browser to position the caption text to the "left", "right", "top" or "bottom". The default is to place the caption at the bottom of the figure. Text flow will occur around the figure and caption, leaving a suitable gully. The browser will ignore this attribute if there is insufficient room for the requested placement.

## Pixel-base or Scaled Coordinates

The upper left of the figure is designated as x,y = (0, 0), with x increasing across the page, and y down the page. If points are given in real numbers, the lower right is taken as being (1.0, 1.0), otherwise with integer values, the coordinates are assumed to be in pixels[8]. Note that using scaled coordinates is much safer, especially for graphics! The extent of the image in pixels may change, e.g. as a result of format negotiation with the server, and by retrieving images with lower resolution when network performance is poor.

## Active areas

The *ismap* attribute causes the browser to send mouse clicks on the figure, back to the server using the selected coordinate scheme. The mouse button-up event is sent with the URL formed by adding "?x,y" as a suffix to the URL for the current document. You can also designate rectangular regions of interest in the picture by holding the mouse button down while dragging the mouse. The browser should show a rubber band outline for the rectangle defined by the current location of the mouse pointer and the point at which the mouse button was pressed. The region is named by taking the current URL and adding the suffix: "?x1,y1;x2,y1;x2,y2"[9], where (x1, y1) and (x2, y2) define the points at which the mouse button went down and came up, respectively. The *ismap* mechanism is relatively slow, but makes sense when the active regions change their boundaries over time, e.g.

```
<fig ismap src="weather.gif">Click on your area for todays weather</fig>
```

You can also designate arbitrary areas of the figure as hypertext links. Mouse clicks are handled locally, and the browser can provide visual clues that the pointer is over an active area, for example, by changing the pointer from an arrow to a hand symbol, or highlighting the area in some way.

Active areas are defined with the FIGA tag. This has two attributes:

| | |
|---|---|
| **href** | A URL specifying the link to traverse when clicked (required) |
| **area** | Defines a polygonal[10] area as a list of points: "x1, y1; x2, y2; ..." (optional) |

The *area* attribute lists a sequence of points defining a polygon. Closure is ensured by joining the last point in the list to the first (i.e. a triangular area is defined with a list of 3 points).When the *area* attribute is missing, the whole of the picture is assumed. Polygons may be non-convex or even intersect themselves, thereby complicating the definition of what is enclosed by the polygon. Holes should be excluded. Note that active areas defined with FIGA take precedence over the *ismap* mechanism.

## Overlays

The FIGT tag allows you to position text and image overlays on top of the figure, e.g.

```
<fig src="map.giff">
  <figt at="0.2, 0.3" framed>A text overlay</figt>
  The figure caption
</fig>
```

---

[8]This mechanism was designed to be backwards compatible with the *ismap* feature as used with IMG in HTML, and as a consequence forces the choice of y increasing down rather than up the page. A simple test to distinguish the two schemes is to check if the "." character occurs anywhere in the list of points.

[9]This definition is intended to allow future extension to arbitrary polygons, and hence is chosen to be directly compatible with the *area* attribute of the FIGA tag.

[10]The code for hit testing polygons is tricky, but quite fast. A public domain version of the code would be helpful.

The overlay can contain a wide variety of elements including text, images (IMG), lists and tables. Figures shouldn't be nested. Any hypertext links in the overlay text will take precedence over the *href* attribute in FIGT. The following attributes are permitted:

**idref** Names the FIG element to which this overlay applies. This is only relevant when overlays are held separately as a form of annotation. This attribute then allows the annotation to be correctly merged back into the document.

**at** The upper left of the overlay, relative to the figure.

**width** As a fraction of the figure, e.g. width="0.3". This allows you to limit the lengths of wrapped text lines. The vertical extent is then determined automatically.

**framed** Directs the browser to draw a frame around the overlay and to colour in the background in some way.

**href** Allows you to make the overlay into a hypertext button.

# Tables

Tables are defined with the TBL tag. Cells are designated as being headers or data. You can join adjacent cells, e.g. to define a header spanning two columns.

An Example of a Table

| | average | | other |
|---|---|---|---|
| | height | weight | category |
| males | 1.9 | 0.003 | yyy |
| females | 1.7 | 0.002 | xxx |

This is defined by the markup:

```
<tbl border>
   <tt top> An Example of a Table
   <th rowspan=2> <th colspan="2"> average <th> other <tr>
   <th> height <th> weight <th> category <tr>
   <th align=left> males <td> 1.9 <td> .003 <td> yyy <tr>
   <th align=left> females <td> 1.7 <td> .002 <td> xxx
</tbl>
```

The *border* attribute for TBL directs the browser to draw borders. The *compact* attribute is used when you want the table to appear in a smaller size.

The optional <tt> tag defines a title. By default (i.e. when *top* is missing) this should be positioned below the table. The <th> and <td> tags define header or data cells respectively. The <tr> tag acts as a separator between rows. In the example, you can see that the first header in each of the first two rows is void.

TH, and TD all have the same permitted attributes:

**colspan** Columns spanned by this cell, see example

**rowspan**[11] Rows spanned by this cell, see example

**align=left** Left justify the cell's content

**align=center** Center justify the cell's content

**align=right** Right justify the cell's content

[11]This is tricky to handle. The parser should carry a spanned cell over to the next row, the definition of which should miss out the spanned cell, i.e. the next row will have one fewer explicit cell definitions.

By default, headers are centered, while other cells are left justified. If practical, browsers should be smarter than this, e.g. if all the cells in a column are shorter than the column header, then indent the cells to make them appear under the middle of the header.

Browsers need to carry out a pre-parse (e.g. when sizing the vertical scroll bar) in order to determine the number of columns and their widths. The following guidelines may be useful:

- There is no need to declare empty cells at the end of a row, so the number of columns for the table is given by the row with the most columns.
- Restricting text to a fixed pitch font may simplify matters.
- If a column only contains numbers or empty cells then align on units and set width to the maxium precision needed (before and after decimal point, allowing for an exponent). This rule also applies when currency symbols are used.
- Otherwise set column width to the minimum of a threshold width and the maximum text length for all cells in the column. Text is left aligned and wrapped if it exceeds the chosen column width.

The threshold column width can be set according to the number of columns and the width of the display window. It is also necessary to take the column headers into account in this process. Header text wraps to the next line if the column is too narrow. Browsers will by default center the header in the column.

A complication occurs when a header or data cell spans more than one column, as specified by the *s* attribute. This can be used to give complex headers which share a header between columns followed by individual headers on the next line.

Vertical gaps can be introduced with the <tb> element - this inserts 1/2 line space into the next row. Header and Data rows can be intermixed. Authors can use alternate header and data rows when the rows alternate between text and numbers. The vertical alignment of numbers only applies to data fields.

Tables which don't fit into this model should be defined as figures using an external format, e.g. Postscript, TeX or Computer Graphics Metafile.

## Forms

A document can include one or more forms. Each form is defined by a FORM element, which contains a number of input fields laid out with normal and preformatted text, lists and tables. The browser should manage the input focus, e.g. with the tab key and mouse clicks. The Return key can be used to mean that the user has filled in the form and wants the appropriate action to be taken. Browsers may also display "Accept" and "Cancel" buttons as part of the document (or perhaps on another part of the browser). Note that forms shouldn't be nested.

The action to be taken is specified by the *action* attribute of the FORM tag. If missing the URL for the current document is assumed. This attribute uses a URL to specify a server to query, or an email address to send the form to. When sending the form to a server as a query, the form's contents are encoded as a property list (see definition of the INPUT tag). The precise encoding is dependent on the HTTP protocol and defined in [Berners-Lee 93c][12]. When the form is to be mailed, it is first converted into plain text, closely resembling the appearence on the screen. You can include multiple RFC 822 mail headers with the MH tag. The *hidden* attribute may be used to hide the headers when browsing the document. The following is an example of a simple questionaire:

```
<form action="mailto:www_admin@info.cern.ch">
<mh hidden>
  Subject: WWW questionaire
</mh>
Please help us to improve the World Wide Web by filling in the
following questionaire:
<p>
Your organisation? <input name="org" size="48">
```

[12]This and the *ismap* feature rely on the forthcoming definition of HTTP as an official Internet standard.

```
<p> commercial? <input name="commerce" type="checkbox">
How many users? <input name="users" type="int">
<p> Which browsers do you use?
<ol compact>
<li> X Mosaic <input name="browsers" type="checkbox" value="xmosaic">
<li> Cello <input name="browsers" type="checkbox" value="cello">
<li> Viola <input name="browsers" type="checkbox" value="viola">
<li> Others? <input name="other browsers" size="48x4">
</ol>
A contact point for your site: <input name="contact" size="48">
<p>Many thanks on behalf of the WWW central support team.
</form>
```

## Floating Panels

The PANEL tag can be used to define panels or boxes which are free to float with respect to the standard flow of text. These are often used in magazine articles for asides on background material. The panel is typically shown with a distinctive background colour and border. The lay out software positions the panel to coincide with the page boundaries in printed media. For on-line use, panels can be rendered as pop-up windows. The body of the panel can be defined by a link to a separate document or included in the current document.

Another application of the panel tag is for annotations by one or more reviewers. The annotations can be held separately e.g. with an annotation server and subsequently retrieved and merged with the document. The annotation server returns a list of annotations in response to being queried with a URL or URN. This list can be expressed as a multi-part MIME message with the author and date passed as RFC 822 headers for each annotation.

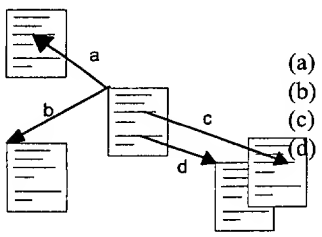
The following optional attributes are permitted with the <panel> tag:

| | |
|---|---|
| **id** | An identifier, unique to this document, which can be used as a destination in a hypertext link. |
| **at** | An identifier elsewhere in this document. The panel mustn't be placed before this point. (Defaults to the current position if the *at* attribute is missing). |
| **role** | This may be used to clarify the role of the panel, e.g. as an "annotation" or an "aside". |
| **href** | This attribute allows authors to fill the panel from a separate document, as specified by a URL. Note that the matching end tag: </panel> is always needed. |

The text contained by the panel element can include any of the markup elements and looks like a separate document (panels themselves can't be nested). If the *href* attribute is used the text delimited by <panel> ... </panel> may be used as the caption for a pop-up. The *at* attribute allows you to include the panel definition at a convenient point in the HTML+ document, rather than interrupting the main flow of the document.

## More on Links

Before describing the details of how links are represented in HTML+ it is worth looking more generally at the nature of hypertext links. First a terminological point: a *node* is the atomic unit for information retrieval, while *documents* may consist of one or more nodes, perhaps arranged as a hierarchy. A node may even be shared between several documents. Hypertext links start and end on nodes or anchors, where anchors specify portions of nodes, e.g. a paragraph or list.




The diagram illustrates the basic possibilities:

(a) Link from a node to an anchor
(b) Link from a node to a node
(c) Link from an anchor to another anchor
(d) Link from an anchor to a node

Links in HTML+ are represented with the LINK and A tags. The LINK tag is used for cases (a) and (b), while the A tag is used for cases (c) and (d). These links are held in the source node only, so there is a risk that the destination may have disappeared. Organisations can manage this risk by long term support for a few well published nodes (servers can use redirection to hide internal name changes for these nodes). Links to other subsidiary nodes are at higher risk. This structured approach allows people to become familiar with the major routes through the web, without needing to worry about the minor routes.

In most cases URLs and URNs explicitly specify a node/anchor. The nodes may be explicit files or generated as the result of some process invoked by the server, e.g. a hypertext listing of a directory or a list of matches for a given search string. The search string can be explicitly encoded as part of a link, or dynamically defined by the user (see the ISINDEX tag, as described later on).

Links may be held separately from the source and destination nodes[13]. This is particularly appropriate for annotations and discussion groups. For example, consider making an annotation on a document held by a server located far away in another organisation. You could take a local copy and directly annotate it, but this is only appropriate for private use. The remote server might even support a protocol to add your annotations in place. More likely though, you will have to use an annotation server. This mechanism can be used to obtain a copy of the document with the annotations inserted as hypertext links and shown as pop-ups or separate documents.

## Context Dependent Links

For discussion groups, responses are made asynchronously, and include one or more references to other articles. In this situation, context dependent links are appropriate. The resolution to an explicit node can be carried out by either the client or server. The former approach is often appropriate, but requires special support in the browser, e.g. for network news and nntp.

Context dependent links are also useful for links to the table of contents for documents consisting of multiple nodes, when some of the nodes also appear in other documents. The appropriate table of contents for a given node will depend on which document is currently being viewed. In this case, the context will depend on how the current node was reached.[14] This is quite simple to track if the links from the table of contents are differentiated from cross reference links.

Hypertext paths are recommended routes through a set of nodes, and generally shown by next and previous buttons on a toolbar. Paths can be defined using explicit links in a node, or held separately in another node. The latter case once again, depends on the context. Paths and tables of contents all fall under the general category of navigating around a hierarchy of nodes forming a document too large or unwieldy to be held in a single node.

## Types of Links

There are several motivations for differentiating between types of links:

| | |
|---|---|
| **how it is viewed** | The potential to show different cues depending on the type and size of the node to be retrieved. If this information is explicitly stated as part of the link, there is a chance that it will become out of step with the linked node. |
| **what happens** | Whether the linked document replaces the current one, or appears in a new window, or as a pop-up overlay on top of the current one. |
| **printed appearance** | Whether links are treated as references, footnotes or as separate sections |
| **effect on context** | After traversing the link, will there be implicit values for the table of contents, and hypertext path etc? |

[13]These correspond to HyTime's *ilink* architectural form.

[14]This is more general than deriving the role of the link from that of the node alone.

## Link Attributes

The A tag has the following attributes:

| | |
|---|---|
| **id** | An identifier unique to this document which can act as a hypertext anchor |
| **name** | The same as *id* and included for backwards compatibility with HTML. New documents should use the *id* attribute for consistency with the other tags. |
| **href** | The URL or URN identifying the destination of the link. |
| **role** | A string giving the role of the link, e.g. role="partof" or "annotation" |
| **effect** | A string defining how the linked node is shown: "replace", "new", "overlay", with the default effect of replacing the current document. |
| **print** | How should the link be printed: "reference", "footnote" and "section", defaulting to "reference" (i.e. a footnote stating the link's URL). |
| **title** | The title to show when otherwise undefined for the node. |
| **type** | The MIME content type for the linked node for use with presentation cues. |
| **size** | The size in bytes for the linked node. This allows the browser to show a gauge indicating progress in retrieving long documents or images etc. |

The LINK tag has only the *href* and *role* attributes.

The *role* attribute is appropriate when context dependent properties such as *table of contents* (toc) are implied for the linked node, e.g. if the current node is a toc (as defined by the *html* or *group* tags) and the link has the role "partof", then the current node should act as the *toc* for the linked node. This property propagates down "partof" links, but not normal links. The *next* and *prev* properties are given by the sequence of "partof" links in the parent node. The *parent* property is only defined if the current node was reached via a "partof" link.

The LINK tag is used to express these properties in an explicit form, e.g.

```
<LINK href="toc.html" role="toc">
```

The recommended property names are: (in upper or lower case)

| | |
|---|---|
| toc | Table of contents for current node. |
| next | The next node in a hypertext path. |
| prev | The previous node in a hypertext path. |
| parent | The next level up in the hierarchy. |
| index | A searchable index appropriate to this node. |
| style | The style sheet appropriate to this node. |

Style sheets provide a way for authors to express their detailed preferences for fonts, and layout, whether for the screen or when the node is printed out. A possible format is given in [Raisch 93].

The *effect* attribute is a hint and may be disregarded by browsers. It allows you to click on an image and to see a linked movie as an overlay at the same position. The browser tries to position the overlay at the same origin as the link. In some cases, the linked node is a description of the current node. By including `effect="new"`, the linked node will appear in a new window so that users can see both nodes at the same time. This hint should be used sparingly!

The *print* attribute makes it practical to print nodes along with relevant linked nodes. By default each link appears as a footnote stating the link's URL. Short nodes can be included in their entirety as footnotes, and longer ones as sections in their own right. This approach could be extended in future, to reorder the sequence of nodes from that defined by the position of the links in the source node, and to control the level that nodes appear as, e.g. chapter, section or subsection.

The *title* attribute is useful for nodes without titles of their own, e.g. Gopher menus. The *type* attribute can be used to show cues for the node type, e.g. iconic decorations[15]. The *size* attribute allows browsers to show a gauge on how much of a document has been retrieved at a any time. These attributes are liable to get out of step with the target node, and should be treated as hints only.

## Groups

The GROUP tag allows you to define arbitrary groups, e.g. books, chapters, and sections. The *role* attribute is used to name the logical role of the group. You can use most markup elements inside a group element, including group itself. The *inset* attribute is a rendering hint to inset the left margin. Using the A tag with role="partof" allows you to designate a node as being included within the group, allowing hierarchies of groups which cross multiple nodes. See previous discussion of how properties are propagated.

Groups offer opportunities for presenting and searching documents at different levels of abstraction. For example, you might first describe a book by its title, author, publisher and ISBN number. The next level down could add a cover illustration together with a summary of the book's contents, some comments by reviewers and a short biography of the author. This would allow a list of books to be presented in an iconic form using a miniature version of the "cover page". Publishers could include copyright and other details in a standard place.

## Change Bars

Authors can indicate a part of a document has been changed using the CHANGED tag. This may appear anywhere that normal text is allowed (as designated by the entity reference `%text;` in the DTD). This tag signals the beginning or end of changes, which should be rendered by a vertical bar in the left margin. The tag can have one (but not both) of the following attributes:

**id** An identifier unique to the current document, which can also be used as a a destination for hypertext links. This signals the beginning of changes, e.g. `<changed id=z34>`.

**idref** This must be an identifier matching the preceding changed element. It signals the end of changes. Note that you mustn't have both *id* and *idref* together, e.g. `<changed idref=<z34>`.

## Notes

The NOTE tag allows authors to name an arbitary portion of a document. It can be used much more freely than the A tag which is restricted in the markup it can contain. The NOTE tag is intended for delimiting the portion of a document associated with an annotation. The annotation itself is linked via the *href* attribute or expressed as a panel element, whose *at* attribute names this note. The latter allows several annotations to apply to the same place.

The permitted attributes are:

**id** An identifier unique to the current document, which can also be used as a a destination for hypertext links. This signals the beginning of the note, e.g. `<note id=z34>`.

**href** May be used to directly specify an associated annotation with a URL.

**idref** This must be an identifier matching the preceding note element. It signals the end of changes. Note that you mustn't have both *id* and *idref* together, e.g. `<note idref=z34>`.

---

[15]The appropriate cue might also depend on the role of the link, e.g. for annotations browsers could show an icon of a drawing pin (as in attaching a note to a pin board). The colour of the pin could then vary according to the media type of the annotation.

## Miscellaneous Tags

The remaining tags must appear at the start of the node like TITLE and LINK. They describe properties which apply to the node as a whole.

### The HTML tag.

This is intended to provide short informal classifications for use in cataloging documents held by HTTP servers. The *role* attribute identifies the purpose of the node, for example `<html role="home page">`. Another common role is "toc" for table of contents. See previous discussion of link attributes.

### The ISINDEX tag

This specifies that the URL designated with the *href* attribute is searchable (defaults to this document's URL). Browsers should allow users to enter a search string of one or more keywords. When the Return key is pressed the search string is appended to the designated URL, after a "?" character and sent to the server specified by the URL. Certain characters should be escaped as specified by the standard URL syntax, for example, the space character is mapped to "+". The newer HTTP protocol offers an alternative means for specifying that documents are searchable. See [Berners-Lee 93c] for details.

### The NEXTID tag

This is used by browsers that automatically generate identifiers for anchor points. It specifies the next identifier to use, to avoid confusion with old (deleted) values, e.g. `<nextid n="id56">`. The identifier should take the form of zero or more letters followed by one or more digits. The numeric suffix should be incremented to generate successive identifiers.

### The BASE tag

The *href* attribute gives the full URL of the document, and is added by the browser when the user makes a local copy. Keeping the original URL in a local copy is essential when subsequently viewing the copy as it allows relative URLs in the document to be resolved to their original references.

Note that one motivation for using relative URLs is to allow a group of documents to be copied without the need to alter any links between them. In this case, the BASE tag is inappropriate, since it would cause links to be intepreted as being to the original documents rather than their copies.

### The HEAD and BODY tags

The HEAD tag can be used to delimit properties which apply to the document as a whole, and if used, must be present at the start of the document, followed by the BODY tag which then delimits the rest of the document.

## Acknowledgements

I would like to thank the many people on the *www-talk* mailing list who have contributed to the design of HTML+ and to the management of HP Labs for their support during this work.

David Raggett, Hewlett Packard Laboratories, July 1993.

---

*Email*: dsr@hplb.hpl.hp.com, *Phone*: +44 272 228046

# Appendix I - The HTML+ DTD

```
<!SGML  "ISO 8879:1986"
--
      Document Type Definition for the HyperText Markup Language
      Plus for use with the World Wide Web application (HTML+
      DTD).
      NOTE: This is a definition of HTML+ with respect to
      SGML, and assumes an understanding of SGML terms.
--
CHARSET
      BASESET     "ISO 646:1983//CHARSET
                  International Reference Version (IRV)//ESC 2/5 4/0"
      DESCSET     0     9     UNUSED
                  9     2     9
                  11    2     UNUSED
                  13    1     13
                  14    18    UNUSED
                  32    95    32
                  127   1     UNUSED
      BASESET     "ISO Registration Number 100//CHARSET
                  ECMA-94 Right Part of Latin Alphabet Nr. 1//ESC 2/13
      4/1"
      DESCSET     128   32    UNUSED
                  160   95    32
                  255   1     UNUSED

CAPACITY    SGMLREF
            TOTALCAP    150000
            GRPCAP              150000

SCOPE       DOCUMENT
SYNTAX
      SHUNCHAR CONTROLS       0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18
                        19 20 21 22 23 24 25 26 27 28 29 30 31 127 255
      BASESET     "ISO 646:1983//CHARSET
            International Reference Version (IRV)//ESC 2/5 4/0"
      DESCSET     0     128     0
      FUNCTION    RE            13
                  RS            10
                  SPACE         32
                  TAB SEPCHAR   9
      NAMING            LCNMSTRT ""
                  UCNMSTRT ""
                  LCNMCHAR ".-"
                  UCNMCHAR ".-"
                  NAMECASE    GENERAL YES
                              ENTITY  NO
      DELIM       GENERAL     SGMLREF
                  SHORTREF    SGMLREF
      NAMES       SGMLREF
      QUANTITY    SGMLREF
                  NAMELEN     34
                  TAGLVL            100
                  LITLEN            1024
                  GRPGTCNT    150
                  GRPCNT            64

FEATURES
      MINIMIZE
            DATATAG  NO
            OMITTAG  NO
            RANK     NO
            SHORTTAG NO
      LINK
            SIMPLE   NO
```

```
            IMPLICIT NO
            EXPLICIT NO
      OTHER
            CONCUR      NO
            SUBDOC      NO
            FORMAL      YES
      APPINFO     NONE
>

<!DOCTYPE HTMLPLUS [

<!-- DTD for HTML+

Markup minimisation should be avoided, otherwise the default
<!SGML> declaration is fine.

Browsers should be forgiving of markup errors.

Common Attributes:

   id       the id attribute allows authors to name elements such as
            headers and paragraphs as potential destinations for links.
            Note that links don't specify points, but rather extended
            objects.

   index    allows authors to specify how given headers etc should be
            indexed as primary or secondary keys, where "/" separates
            primary from secondary keys, ";" separates multiple entries

-->

<!-- ENTITY DECLARATIONS

  <!ENTITY % foo "X | Y | Z"> is a macro definition for parameters and in
  subsequent statements, the string "%foo;" is expanded to "X | Y | Z"

  Various classes of SGML text types:

    #CDATA   text which doesn't include markup or entity references

    #RCDATA  text with entity references but no markup

    #PCDATA  text occurring in a context in which markup and entity
             references may occur.
-->

<!ENTITY % URL "CDATA" -- a URL or URN designating a hypertext node -->

<!ENTITY % text "#PCDATA|A|IMG|EM|EMBED|INPUT|BR|CHANGED|NOTE">

<!ENTITY % paras "P|PRE|FIG">

<!ENTITY % lists "UL|OL|DL">

<!ENTITY % misc "HR|TBL|FORM|PANEL|GROUP">

<!ENTITY % heading "H1|H2|H3|H4|H5|H6">

<!ENTITY % table "%text;|P|%heading;|%lists;">

<!ENTITY % main "%heading;|%misc;|%lists;|%paras;|%text;">

<!ENTITY % setup "(TITLE? & HTML? &ISINDEX? & NEXTID? & LINK* & BASE?)">

<!--

<!ELEMENT tagname - - CONTENT> elements needing closing tags

<!ELEMENT tagname - O CONTENT> elements without closing tags

<!ELEMENT tagname - O EMPTY> elements without content or closing tags

   The content definition is:

      a)    an entity definition as defined above

      b)    a tagname
```

```
      c)    (brackets enclosing the above)
   These may be combined with the operators:
      A*    A occurs zero or more times
      A+    A occurs one or more times
      A|B   implies either A or B
      A?    A occurs zero or one times
      A,B   implies first A then B
-->
<!ELEMENT HTMLPLUS O O ((HEAD, BODY) | ((%setup;), (%main;)*))>
<!ELEMENT HEAD - - (%setup;)>
<!ELEMENT BODY - - (%main;)*>
<!-- Document title -->
<!ELEMENT TITLE - - (#PCDATA | EM)+>
<!ATTLIST TITLE
      id       ID      #IMPLIED -- link destination --
      index    CDATA   #IMPLIED -- entries for index compilation -->
<!-- Document role for cataloging documents held by servers -->
<!ELEMENT HTML - O (EMPTY)>
<!ATTLIST HTML role CDATA #IMPLIED -- home page, index, ... -->
<!-- Floating panel which can be moved around relative to the normal text
flow. Often rendered with a different background and possibly framed. The
panel can be anchored to a named point in the document as specified by
the AT attribute. The panel may be placed at that point or after, but not
before.
-->
<!ELEMENT PANEL - - (TITLE?, (%main;)*)>
<!ATTLIST PANEL
      id       ID      #IMPLIED -- defines link destination --
      at       IDREF   #IMPLIED -- anchor point --
      role     CDATA   #IMPLIED -- annotation/aside
      index    CDATA   #IMPLIED -- entries for index compilation -->
<!ELEMENT HR - O EMPTY -- Horizontal Rule -->
<!-- Document headers -->
<!ELEMENT (%heading;) - - (#PCDATA | EM)+>
<!ATTLIST (%heading;)
      id       ID      #IMPLIED -- defines link destination --
      index    CDATA   #IMPLIED -- entries for index compilation -->
<!-- logical emphasis with optional style hints -->
<!ELEMENT EM - - (%text;)*>
<!ATTLIST EM
   role   CDATA  #IMPLIED -- semantic category e.g. CITE --
   b      (b)    #IMPLIED -- render in bold font --
   i      (i)    #IMPLIED -- render in italic font --
   u      (u)    #IMPLIED -- underline text --
   tt     (tt)   #IMPLIED -- render in typewriter font --
   tr     (tr)   #IMPLIED -- render in serif (Times Roman) font --
   hv     (hv)   #IMPLIED -- render in sans serif (Helvetica) font --
   sup    (sup)  #IMPLIED -- superscript --
   sub    (sub)  #IMPLIED -- subscript --
   index  CDATA  #IMPLIED -- entries for index compilation -->
```

```
<!-- Paragraphs with different roles and optional style hints
     Note that paragraphs act as containers for the following text -->

<!ELEMENT P - O (%text;)+>

<!ATTLIST P
   id       ID          #IMPLIED -- link destination --
   role     CDATA       #IMPLIED -- semantic role --
   align    CDATA       #IMPLIED -- left, center or right --
   indent   (indent)    #IMPLIED -- indented margins --
   index    CDATA       #IMPLIED -- entries for index compilation -->

<!ELEMENT BR - O EMPTY -- line break in normal text-->

<!-- Preformatted text with fixed pitch font, respecting original spacing
and newlines. Authors can also request proportional fonts. Further
control is possible with EM, and TAB -->

<!ELEMENT PRE - - (TAB|%text;)+>

<!ATTLIST PRE
   id       ID       #IMPLIED -- link destination --
   role     CDATA    #IMPLIED -- various styles --
   tr       (tr)     #IMPLIED -- serif (Times Roman) font --
   hv       (hv)     #IMPLIED -- sans serif (Helvetica) font --
   width    NUMBER   #IMPLIED -- e.g. 40, 80, 132 --
   index    CDATA    #IMPLIED -- entries for index compilation -->

<!ELEMENT TAB - O EMPTY>

<!ATTLIST TAB
   at  NUMBER  #IMPLIED -- position measured in widths of a capital M --
   align  (left|center|right|decimal)  left -- tab alignment -->

<!-- Lists which can be nested -->

<!ELEMENT OL - - (LI | UL | OL)+ -- ordered list -->

<!ATTLIST OL
   id       ID         #IMPLIED
   compact  (compact)  #IMPLIED
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!ELEMENT UL - - (LI | UL | OL)+ -- unordered list -->

<!ATTLIST UL
   id       ID         #IMPLIED      -- link destination --
   compact  (compact)  #IMPLIED      -- reduced interitem spacing --
   narrow   (narrow)   #IMPLIED      -- narrow perhaps multi columns --
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!-- List items for UL and OL lists -->

<!ELEMENT LI - O (P|%text;)+>

<!ATTLIST LI
   id       ID         #IMPLIED
   src      %URL;      #IMPLIED -- icon for use in place of bullet --
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!-- Definition Lists (terms + definitions) -->

<!ELEMENT DL - - (DT,DD)+ -- DT and DD *MUST* be paired -- > <!ATTLIST DL
   id       ID         #IMPLIED
   compact  (compact)  #IMPLIED
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!ELEMENT DT - O (%text;)+ -- term text -- >

<!ELEMENT DD - O (P|UL|OL|%text;)+ -- definition text -- >

<!ATTLIST (DT|DD)
   id        ID        #IMPLIED
   index     CDATA     #IMPLIED -- entries for index compilation -->
```

```
<!-- Tables with titles and column headers, e.g.
   <tbl border>
      <tt> An Example of a Table
      <th> <th s="2"> average <th> other <tr>
      <th> <th> height <th> weight <th> category <tr>
      <td> males <td> 1.9 <td> .003 <td> yyy <tr>
      <td> females <td> 1.7 <td> .002 <td> xxx
   </tbl>
-->

<!ELEMENT TBL - - (TT?, (TH|TD|TR|TB)*) -- mixed headers and data -->

<!ATTLIST TBL
   id       ID         #IMPLIED
   compact  (compact)  #IMPLIED      -- if present use compact style --
   border   (border)   #IMPLIED      -- if present draw borders --
   index    CDATA      #IMPLIED       -- entries for index compilation -->

<!ELEMENT TT - O (%text;)+ -- table title -->

<!ATTLIST TT top (top) #IMPLIED -- place title above table -->

<!ELEMENT TH - O (%table;)* -- a header cell -->

<!ATTLIST TH
   colspan  NUMBER     1          -- columns spanned --
   rowspan  NUMBER     1          --. rows spanned --
   align    CDATA    #IMPLIED     -- left, center or right -->

<!ELEMENT TD - O (%table;)* -- a data cell -->

<!ATTLIST TD
   colspan  NUMBER     1          -- columns spanned --
   rowspan  NUMBER     1          --. rows spanned --
   align    CDATA    #IMPLIED     -- left, center or right -->

<!ELEMENT TR - O EMPTY -- row separator -->

<!ELEMENT TB - O EMPTY -- vertical break of 1/2 line spacing -->

<!-- Forms composed from input fields and selection menus

These elements define fields which users can type into or select with
mouse clicks. The browser should manage the input focus e.g. with the
tab/shift tab keys and mouse clicks.

The enter/return key is then taken to mean the use has filled in the form
and wants the apppropriate action taken:
   - send as query/update to WWW server
   - email/fax to designated person

The action is specified as a URL, e.g. "mailto:dsr@hplb.hpl.hp.com You
can specify additional mail headers with the MH tag:

<MH>Subject: Please add me to tennis tournament</MH>

Each FORM should include one or more INPUT elements which can be layed
out with normal and preformatted text, lists and tables.
 -->

<!ELEMENT FORM - - (MH, (%main;)*)>

<!ATTLIST FORM
   id       ID         #IMPLIED
   action   %URL;      #IMPLIED
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!ELEMENT MH - - CDATA -- one or more RFC 822 header fields -->

<!ATTLIST MH hidden (hidden) #IMPLIED -- hide the mail headers from view -->
```

```
<!-- INPUT elements should be defined within a FORM element.

Users can alter the value of the INPUT element by typing or clicking with
the mouse. Use radio buttons for selecting one attribute value from a set
of alternatives. In this case there will be several INPUT elements with
the same name. Attributes which can take multiple values at the same time
should be defined with checkboxes: define each allowed value in a
separate INPUT element but with the same attribute name. For checkboxes
and radio buttons, the value doesn't change, instead the state of the
button shown by the presence or absence of the checked attribute in each
element.

The size attribute specifies the size of the input field as appropriate
to each type. For text this gives the width in characters and height in
lines (separated by an "x"). For numbers this gives the maximum
precision.

-->

<!ELEMENT INPUT - O EMPTY>

<!ATTLIST INPUT
   name     CDATA     #IMPLIED -- attribute name (may not be unique) --
   type     CDATA     #IMPLIED --TEXT,URL,INT,FLOAT,DATE,CHECKBOX,RADIO--
   size     CDATA     #IMPLIED -- e.g."32x4" for multiline text --
   value    CDATA     #IMPLIED -- attribute value (altered by user) --
   checked  (checked) #IMPLIED -- for check boxes and radio buttons --
   disabled(disabled)   #IMPLIED    -- if grayed out --
   error    (error)   #IMPLIED -- if in error -->

<!-- Embedded Data

You can embed information in a foreign format into the HTML+ document.
This is very convenient for mathematical equations and simple drawings.
Images and complex drawings are better specified as linked documents
using the FIG or IMG elements.

Arbitrary 8 bit data is allowed but any occurrences of the following
chars must be escaped as shown:

      "&"     by      "&amp;"
      "<"     by      "&lt;"
      ">"     by      "&gt;"
The browser can pipe such data thru filters to generate the corresponding
pixmap The data format is specified as a MIME content type, e.g.
"text/eqn"

It would have been nice to use the NOTATION type in place of CDATA, but
this doesn't
appear to be practical with externally specified content type names.
-->

<!ELEMENT EMBED - - (RCDATA)>
<!ATTLIST EMBED
   id       ID        #IMPLIED
   type     CDATA     #IMPLIED  -- mime content type --
   index    CDATA     #IMPLIED -- entries for index compilation -->

<!-- Figures

The image/drawing is specified by a URL or as embedded data for simple
drawings. The element's text serves as the caption. Use the emphasis with
style = "credits" to record photo credits etc.

-->
```

```
<!ELEMENT FIG - - (EMBED?, FIGD?, (FIGA|FIGT)*, (%text;)*)>

<!ATTLIST FIG
   id       ID         #IMPLIED
   align    CDATA      #IMPLIED -- position: left, right or center --
   cap      CDATA      #IMPLIED -- caption at left, right, top, bottom --
   noflow   (noflow)   #IMPLIED -- disables text flow --
   ismap    (ismap)    #IMPLIED -- server can handle mouse clicks/drags --
   src      %URL;      #IMPLIED -- link to image data --
   index    CDATA      #IMPLIED -- entries for index compilation -->

<!ELEMENT FIGD - - (%table;) -- figure description -->

<!-- Figure anchors designate polygonal areas on the figure which can be
clicked with the mouse. The default area is the whole of the figure. This
mechanism interprets mouse clicks locally, and browsers can choose to
highlight the designated area (or change the mouse sprite) when the mouse
is moved over the area.

Note that polygons may be non-convex or even intersect themselves,
thereby complicating the definition of what is enclosed by the polygon.
Holes are excluded.
-->

<!ELEMENT FIGA - O EMPTY>

<!ATTLIST FIGA
   href    %URL;     #REQUIRED -- link to traverse when clicked --
   area    NUMBERS   #IMPLIED -- x1,y1,x2,y2,x3,y3,... -->

<!-- FIGT Text on top of an figure background, or in a colored background
box which sits arbitrarily on top of an figure background. The text can
include headers, lists and tables etc. The width attribute allows you to
limit the width of the text box. The height is then determined
automatically by the browser.

FIGT can also be used to position a graphic on top of a picture using an
IMG element within FIGT. In this case the chromakey attribute may allow
parts of the underlying image to show through.

You can make the whole of the box into a hypertext link. This will act as
if it is underneath any hypertext links specified by the overlay markup
itself.

FIGT can also be used for annotations on figures, and held separately
from the document with the figure to which they apply. In this case, use
the idref attribute to name which figure is appropriate.
-->

<!ELEMENT FIGT - - (%main;)>

<!ATTLIST FIGT
   idref  IDREF    #IMPLIED -- names FIG element if held separately --
   at     NUMBERS  #IMPLIED -- upper left origin for text --
   width  NUMBER   #IMPLIED -- given as fraction of picture --
   framed (framed) #IMPLIED -- framed with coloured background --
   href   %URL;    #IMPLIED -- link to traverse when clicked -->

<!-- inline icons/small graphics
The align attribute defines whether the top middle or bottom of the
graphic and current text line should be aligned vertically

The SEETHRU attribute is intended as a chromakey to allow a given colour
to be designated as "transparent". Pixels with this value should not be
painted. The exact format of this attribute's value has yet to be
defined.


Use the FIG tag for captioned figures with active areas etc.
-->

<!ELEMENT IMG - O EMPTY>
```

```
<!ATTLIST IMG
   src     %URL;   #REQUIRED -- where to get image data --
   align   CDATA   #IMPLIED  -- top, middle or bottom --
   seethru CDATA   #IMPLIED  -- for transparency --
   ismap   (ismap) #IMPLIED  -- send mouse clicks/drags to server -->

<!-- Hierarchical groups for books, chapters, sections etc. -->

<!ELEMENT GROUP - - ((TITLE|LINK*), (%main;)*)>

<!ATTLIST GROUP
   id    ID        #IMPLIED
   role  CDATA     #IMPLIED -- book, chapter, section etc. --
   inset (inset)   #IMPLIED -- rendering hint: indent margins -->

<!-- change bars defined by a matched pair of CHANGED elements:

      <changed id=z34> changed text <changed idref=z34>

This tag can't act as a container, since changes don't respect
the nesting implied by paragraphs, headers, lists etc.
-->

<!ELEMENT CHANGED - O EMPTY>

<!ATTLIST CHANGED -- one of id and idref is always required --
   id       ID      #IMPLIED -- signals start of changes --
   idref    IDREF   #IMPLIED -- signals end of changes -->

<!-- Matched pairs of NOTE elements can be used to delimit text
associated with one or more annotations. The annotation itself can be
linked via an explicit URL or defined as one or more PANEL elements
naming this NOTE.

      <note id=z34> delimited text <note idref=z34>

This tag can't act as a container, since users may select text without
regard to
the nesting implied by paragraphs, headers, lists etc.
-->

<!ELEMENT NOTE - O EMPTY>

<!ATTLIST NOTE -- one of id and idref is always required --
   id       ID      #IMPLIED -- signals start of delimited text --
   href     %URL;   #IMPLIED -- for directly specifying the annotation --
   idref    IDREF   #IMPLIED -- signals end of delimited text -->

<!-- Hypertext Links from points within document nodes -->

<!ELEMENT A - - (#PCDATA | IMG | EM | EMBED)*>

<!ATTLIST A
   id       ID       #IMPLIED -- as target of link --
   name     CDATA    #IMPLIED -- for backwards compatibility with HTML--
   href     %URL;    #IMPLIED -- destination node --
   role     CDATA    #IMPLIED -- role of link, e.g. "partof" --
   effect   CDATA    #IMPLIED -- replace/new/overlay --
   print    CDATA    #IMPLIED -- reference/footnote/section --
   title    CDATA    #IMPLIED -- when otherwise unavailable --
   type     CDATA    #IMPLIED -- for presentation cues --
   size     NAMES    #IMPLIED -- for progress cues -->
```

```
<!-- Other kinds of relationships between documents -->

<!ELEMENT LINK - O EMPTY>

<!ATTLIST LINK
   href     %URL;     #IMPLIED -- destination node --
   role     CDATA     #IMPLIED -- role played, e.g. "toc" -->

<!-- Original document URL for resolving relative URLs  -->

<!ELEMENT BASE - O EMPTY>
<!ATTLIST BASE HREF %URL; #IMPLIED>

<!-- Signifies the document's URL accepts queries -->

<!ELEMENT ISINDEX - O (EMPTY)>
<!ATTLIST ISINDEX href %URL; #IMPLIED -- defaults to document's URL -->

<!-- For use with autonumbering editors - don't reuse ids, allocate next
one starting from this one -->

<!ELEMENT NEXTID - O (EMPTY)>
<!ATTLIST NEXTID N NAME #REQUIRED>

<!-- Mnemonic character entities for 8 bit ANSI Latin-1 -->

<!ENTITY iexcl "&#161;" -- inverted exclamation mark -->
<!ENTITY cent "&#161;" -- cent sign -->
<!ENTITY pound "&#163;" -- pound sign -->
<!ENTITY yen "&#165;" -- yen sign -->
<!ENTITY brvbar "&#166;" -- broken vertical bar -->
<!ENTITY sect "&#167;" -- section sign -->
<!ENTITY copy "&#169;" -- copyright sign -->
<!ENTITY laquo "&#171;" -- angle quotation mark, left -->
<!ENTITY raquo "&#187;" -- angle quotation mark, right -->
<!ENTITY not "&#172;" -- negation sign -->
<!ENTITY reg "&#174;" -- circled R registered sign -->
<!ENTITY deg "&#176;" -- degree sign -->
<!ENTITY plusmn "&#177;" -- plus or minus sign -->
<!ENTITY sup2 "&#178;" -- superscript 2 -->
<!ENTITY sup3 "&#179;" -- superscript 3 -->
<!ENTITY micro "&#181;" -- micro sign -->
<!ENTITY para "&#182;" -- paragraph sign -->
<!ENTITY sup1 "&#185;" -- superscript 1 -->
<!ENTITY middot "&#183;" -- center dot -->
<!ENTITY frac14 "&#188;" -- fraction 1/4 -->
<!ENTITY frac12 "&#189;" -- fraction 1/2 -->
<!ENTITY iquest "&#191;" -- inverted question mark -->
<!ENTITY frac34 "&#190;" -- fraction 3/4 -->
<!ENTITY AElig "&#198;" -- capital AE diphthong (ligature) -->
<!ENTITY Aacute "&#193;" -- capital A, acute accent -->
<!ENTITY Acirc "&#194;" -- capital A, circumflex accent -->
<!ENTITY Agrave "&#192;" -- capital A, grave accent -->
<!ENTITY Aring "&#197;" -- capital A, ring -->
<!ENTITY Atilde "&#195;" -- capital A, tilde -->
<!ENTITY Auml "&#196;" -- capital A, dieresis or umlaut mark -->
<!ENTITY Ccedil "&#199;" -- capital C, cedilla -->
<!ENTITY ETH "&#208;" -- capital Eth, Icelandic -->
<!ENTITY Eacute "&#201;" -- capital E, acute accent -->
<!ENTITY Ecirc "&#202;" -- capital E, circumflex accent -->
<!ENTITY Egrave "&#200;" -- capital E, grave accent -->
<!ENTITY Euml "&#203;" -- capital E, dieresis or umlaut mark -->
<!ENTITY Iacute "&#205;" -- capital I, acute accent -->
<!ENTITY Icirc "&#206;" -- capital I, circumflex accent -->
<!ENTITY Igrave "&#204;" -- capital I, grave accent -->
<!ENTITY Iuml "&#207;" -- capital I, dieresis or umlaut mark -->
<!ENTITY Ntilde "&#209;" -- capital N, tilde -->
<!ENTITY Oacute "&#211;" -- capital O, acute accent -->
<!ENTITY Ocirc "&#212;" -- capital O, circumflex accent -->
<!ENTITY Ograve "&#210;" -- capital O, grave accent -->
```

```
<!ENTITY Oslash "&#216;" -- capital O, slash -->
<!ENTITY Otilde "&#213;" -- capital O, tilde -->
<!ENTITY Ouml "&#214;" -- capital O, dieresis or umlaut mark -->
<!ENTITY THORN "&#222;" -- capital THORN, Icelandic -->
<!ENTITY Uacute "&#218;" -- capital U, acute accent -->
<!ENTITY Ucirc "&#219;" -- capital U, circumflex accent -->
<!ENTITY Ugrave "&#217;" -- capital U, grave accent -->
<!ENTITY Uuml "&#220;" -- capital U, dieresis or umlaut mark -->
<!ENTITY Yacute "&#221;" -- capital Y, acute accent -->
<!ENTITY aacute "&#225;" -- small a, acute accent -->
<!ENTITY acirc "&#226;" -- small a, circumflex accent -->
<!ENTITY aelig "&#230;" -- small ae diphthong (ligature) -->
<!ENTITY agrave "&#224;" -- small a, grave accent -->
<!ENTITY amp "&amp;" -- ampersand -->
<!ENTITY aring "&#229;" -- small a, ring -->
<!ENTITY atilde "&#227;" -- small a, tilde -->
<!ENTITY auml "&#228;" -- small a, dieresis or umlaut mark -->
<!ENTITY ccedil "&#231;" -- small c, cedilla -->
<!ENTITY eacute "&#233;" -- small e, acute accent -->
<!ENTITY ecirc "&#234;" -- small e, circumflex accent -->
<!ENTITY egrave "&#232;" -- small e, grave accent -->
<!ENTITY eth "&#240;" -- small eth, Icelandic -->
<!ENTITY euml "&#235;" -- small e, dieresis or umlaut mark -->
<!ENTITY gt "&#62;" -- greater than -->
<!ENTITY iacute "&#237;" -- small i, acute accent -->
<!ENTITY icirc "&#238;" -- small i, circumflex accent -->
<!ENTITY igrave "&#236;" -- small i, grave accent -->
<!ENTITY iuml "&#239;" -- small i, dieresis or umlaut mark -->
<!ENTITY lt "&lt;" -- less than -->
<!ENTITY ntilde "&#241;" -- small n, tilde -->
<!ENTITY oacute "&#243;" -- small o, acute accent -->
<!ENTITY ocirc "&#244;" -- small o, circumflex accent -->
<!ENTITY ograve "&#242;" -- small o, grave accent -->
<!ENTITY oslash "&#248;" -- small o, slash -->
<!ENTITY otilde "&#245;" -- small o, tilde -->
<!ENTITY ouml "&#246;" -- small o, dieresis or umlaut mark -->
<!ENTITY szlig "&#223;" -- small sharp s, German (sz ligature) -->
<!ENTITY thorn "&#254;" -- small thorn, Icelandic -->
<!ENTITY uacute "&#250;" -- small u, acute accent -->
<!ENTITY ucirc "&#251;" -- small u, circumflex accent -->
<!ENTITY ugrave "&#249;" -- small u, grave accent -->
<!ENTITY uuml "&#252;" -- small u, dieresis or umlaut mark -->
<!ENTITY yacute "&#253;" -- small y, acute accent -->
<!ENTITY yuml "&#255;" -- small y, dieresis or umlaut mark -->


<!-- other entities -->

<!ENTITY ndash "--"  -- En dash -->
<!ENTITY mdash "---" -- Em dash -->

<!ENTITY nbsb " " -- non breaking space -->
<!ENTITY ensp "  " -- En space -->
<!ENTITY emsp "    " -- Em space -->

<!ENTITY shy "-" -- soft hyphen -->


<!-- The END -->

]>
```

# Appendix II - Entity Definitions

The following character definitions conform to widely available 8 bit character sets, e.g. the ANSI Latin-1 character set which is available for both the PC and X11. The corresponding 8-bit character codes are given in the DTD. All definitions conform to the ISO 8879-1986 naming conventions.

| | |
|---|---|
| Æ | capital AE diphthong (ligature) |
| Á | capital A, acute accent |
| Â | capital A, circumflex accent |
| À | capital A, grave accent |
| Å | capital A, ring |
| Ã | capital A, tilde |
| Ä | capital A, dieresis or umlaut mark |
| Ç | capital C, cedilla |
| Ð | capital Eth, Icelandic |
| É | capital E, acute accent |
| Ê | capital E, circumflex accent |
| È | capital E, grave accent |
| Ë | capital E, dieresis or umlaut mark |
| Í | capital I, acute accent |
| Î | capital I, circumflex accent |
| Ì | capital I, grave accent |
| Ï | capital I, dieresis or umlaut mark |
| Ñ | capital N, tilde |
| Ó | capital O, acute accent |
| Ô | capital O, circumflex accent |
| Ò | capital O, grave accent |
| Ø | capital O, slash |
| Õ | capital O, tilde |
| Ö | capital O, dieresis or umlaut mark |
| Þ | capital THORN, Icelandic |
| Ú | capital U, acute accent |
| Û | capital U, circumflex accent |
| Ù | capital U, grave accent |
| Ü | capital U, dieresis or umlaut mark |
| Ý | capital Y, acute accent |
| á | small a, acute accent |
| â | small a, circumflex accent |
| æ | small ae diphthong (ligature) |
| à | small a, grave accent |
| å | small a, ring |
| ã | small a, tilde |
| ä | small a, dieresis or umlaut mark |
| ç | small c, cedilla |
| é | small e, acute accent |
| ê | small e, circumflex accent |
| è | small e, grave accent |
| ð | small eth, Icelandic |

| | |
|---|---|
| ë | small e, dieresis or umlaut mark |
| í | small i, acute accent |
| î | small i, circumflex accent |
| ì | small i, grave accent |
| ï | small i, dieresis or umlaut mark |
| ñ | small n, tilde |
| ó | small o, acute accent |
| ô | small o, circumflex accent |
| ò | small o, grave accent |
| ø | small o, slash |
| õ | small o, tilde |
| ö | small o, dieresis or umlaut mark |
| ß | small sharp s, German (sz ligature) |
| þ | small thorn, Icelandic |
| ú | small u, acute accent |
| û | small u, circumflex accent |
| ù | small u, grave accent |
| ü | small u, dieresis or umlaut mark |
| ý | small y, acute accent |
| ÿ | small y, dieresis or umlaut mark |

In addition, there are some common publishing characters:

| | |
|---|---|
| ¡ | inverted exclamation mark |
| ¢ | cent sign |
| £ | pound sign |
| ¥ | yen sign |
| ¦ | broken vertical bar |
| § | section sign |
| © | copyright sign |
| « | angle quotation mark, left |
| » | angle quotation mark, right |
| ¬ | negation sign |
| ® | circled R registered sign |
| ° | degree sign |
| ± | plus or minus sign |
| ² | superscript 2 |
| ³ | superscript 3 |
| µ | micro sign |
| ¶ | paragraph sign |
| ¹ | superscript 1 |
| · | center dot |
| ¼ | fraction 1/4 |
| ½ | fraction 1/2 |
| ¾ | fraction 3/4 |
| ¿ | inverted question mark |

Finally, there are several special purpose definitions:

| | |
|---|---|
| – | En sized horizontal dash (--) |
| — | Em sized horizontal dash (---) |
|   | Non-breaking space |
|   | En space ( ) |
|   | Em space ( ) |
|  | Soft hyphen |

# Appendix III - Compatibility with HTML

HTML+ browsers should be able to view HTML documents with very little extra code and it is strongly recommended that browsers support both formats. Older HTML browsers will be able to view HTML+ documents which don't contain figures, tables or forms.

## Lists

| HTML | HTML+ |
|---|---|
| <menu> | <ul compact> |
| <dir> | <ul narrow> |

## Emphasis

HTML+ replaces the various tags used by HTML with a single tag. It may be worth changing the name for the emphasis tag in HTML+ from EM to EM, to gain compatibility with this common form. However, using EM might be confused with the typographical term *em* as in em dash (you also get en dash). EM has the merit of being unambigous.

| HTML | HTML+ |
|---|---|
| <tt> | <em tt> |
|  | <em b> |
| <b> | <em b> |
| <i> | <em i> |
| <u> | <em u> |
| <code> |  |
| <samp> |  |
| <kbd> |  |
| <var> |  |
| <dfn> |  |
| <cite> |  |

## Miscellaneous

Some tags which are deprecated in HTML are now obsolete, and should be mapped to preformatted text. This doesn't work quite right as PRE assumes that characters such as "<", ">" and "&" have been replaced by their entity definitions. Browsers should perhaps treat "<" as verbatim unless it forms part of an expected tag. In this way, unescaped occurrences of these three characters will normally display as intended.

| HTML | HTML+ |
|---|---|
| <plaintext> | <pre> |
| <xmp> | <pre> |
| <listing> | <pre> |

The following two tags have been absorbed into the standard mechanism for paragraphs:

| | | |
|---|---|---|
| <address> | becomes | <p role="byline" align="right"> |
| <blockquote> | becomes | <p role="quote"> |

### Notes for Implementors

Please ensure that browsers can tolerate bad markup. In practice, this is quite straightforward to achieve, provided a naive top-down SGML parser is avoided. A forgiving parser should be able to cope with tags in unexpected positions, e.g. the <A> tag bracketing a header[16]. Unknown tags should be simply ignored.

Implementors should endevour to make sure that documents can be scrolled efficiently regardless of their length. Always parsing from the start of the document leads to jerky performance. Two strategies for efficiently scrolling through documents are:

a) Establish regular landmarks throughout the document for which the state of the parse is known. The browser can then work forward from the nearest landmark, when it needs to refresh the screen after a scroll operation. The landmarks need updating when users make changes, while using a WYSIWYG editor.

b) When scrolling up, parse backwards to work out the state at earlier points in the document. This can be done via a combination of skipping back, looking for markup which causes a line break etc. and then parsing forward until the current position, to find the change of state. This can be repeated until the parser reaches a point prior to the new top of the window.

Practical experience has shown the importance of providing cues to users on progress in retrieving documents over the network. These will depend on the protocol, but should show at least how much data has been received at any point. The network connections shouldn't block, and an abort button is essential[17]. It is generally better to avoid displaying the retrieved document in a new window, unless explicitly requested by the user, e.g. by holding down the shift key when clicking the hypertext link.

# References

This is missing the appropriate references to work on the syntax and name service for URNs. The HTTP definition needs updating to cover the encoding of form data (and *ismap* ?).


[Berners-Lee 93a] "*Hypertext Markup Language (HTML)*", Tim Berners-Lee, March 1993.
URL=`ftp://info.cern.ch/pub/www/doc/http-spec.ps`

[Berners-Lee 93b] "*Uniform Resource Locators*", Tim Berners-Lee, January 1992.
URL=`ftp://info.cern.ch/pub/ietf/url4.ps`

[Berners-Lee 93c] "*Protocol for the Retrieval and Manipulation of Textual and Hypermedia Information*", Tim Berners-Lee, 1993.
URL=`ftp://info.cern.ch/pub/www/doc/html-spec.ps`

[Goldfarb 90] "*The SGML Handbook*", Charles F. Goldfarb, Clarendon Press · Oxford.

[Kimber 93] Article in comp.text.sgml newsgroup, 24th May 1993 by Elliot Kimber (drmacro@vnet.almaden.ibm.com),
URL=`news:19930524.152345.29@almaden.ibm.com`

[Raisch 93] "*Style sheets for HTML*", Robert Raisch, June 1993, O'Reilly & Associates
email: `raisch.ora.com`


---

[16]Headers typically cause a line break and leave a vertical gap. If the hypertext link definition is parsed prior to the beginning of the header, the starting position for the button will be in the wrong place - browsers should therefore adjust this position to the beginning of the text.

[17]For X11 on Unix systems, the *select* system call can be used with non-blocking I/O to poll the event queue at regular intervals. The *XtAddInput* call acts as a wrapper around *select* for this very purpose. Users can then continue to view the current document as well as being able to click an abort button (which sets a global variable, polled by the comms software). Be careful to disable unsafe actions, e.g. trying to get a second document while still waiting to get the first (a race hazard).

Photograph Not Found

Attachment B

Posting of Dave Raggett, dsr@hplb.hpl.hp.com, to www-talk@nxoc01.cern.ch (June 14, 1993) (posting to WWW-Talk public mailing list)

("*Raggett II*")

# HTML+ support for eqn & Postscript

*Dave_Raggett <dsr@hplb.hpl.hp.com>*


- **Mail folder:** WWW Talk Apr-Jun 1993 Archives
- **Next message:** Marc Andreessen: "Re: SPaces and Tabs in HTML documents"
- **Previous message:** Tim Berners-Lee: "Re: SPaces and Tabs in HTML documents"
- **Reply:** Torben Noerup Nielsen: "re: HTML+ support for eqn & Postscript"
- **Reply:** Bill Janssen: "Re: HTML+ support for eqn & Postscript"
- **Reply:** Bill Janssen: "Re: HTML+ support for eqn & Postscript"


From: Dave_Raggett <dsr@hplb.hpl.hp.com>
Message-id: <9306140936.AA00563@manuel.hpl.hp.com>
Subject: HTML+ support for eqn & Postscript
To: torben@hawaii.edu, janssen@parc.xerox.com
Date: Mon, 14 Jun 93 10:36:40 BST
Cc: www-talk@nxoc01.cern.ch
Mailer: Elm [revision: 66.36.1.1]

Torben Nielsen says:

> I realize this has come up before, but how about really doing something about
> equation support? There are lots of documents I would like to put into the
> Web, but without support for embedded equations, it's really quite difficult.
> Something simple like eqn support would be great. And eqn shouldn't be all
> that hard to parse either.....

Bill Janssen chips in with:

> And I really like to send encapsulated Postcript in my documents...
> Having ghostscript should make the parsing and layout easy!

Well both of these will be possible with the HTML+ DTD, by using the capability to embed foreign formats inline in the HTML+ source, e.g.

```
<H2>A example of an equation</H2>

<EMBED TYPE="text/eqn">zeta (s) ~=~ sum from k=1 to inf
 k sup -s ~~~ (Re s &gt; 1) </EMBED>
```

The browser identifies the format of the embedded data from the "type" attribute, specified as a MIME content type. Certain characters need to be escaped using entity definitions, e.g. ">" by ">" in the example.

Building in support for a range of formats has the danger of leading to very large programs for browsers. This could be avoided by using a common API for rendering foreign formats, e.g. as functions that take a sequence of bytes and return a pixmap.

Browsers can then be upgraded to display new formats without changing their code at all. All you would need is a way of binding the MIME content type to the function name for that format, e.g. via X resources or a config file. The functions could be implemented as separate programs driven via pipes and stdin/stdout or as dynamically linked library modules (Windows DLLs).

How does that sound?

Dave

p.s. you can also put the foreign data in a separate file referenced by a URL.

Archive - Authorized

C

3838444 - 08/12/04

# Attachment C

# Declaration of David F. Raggett

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

**Declaration Under 37 CFR 1.132**

**DECLARATION OF DAVID St.John RAGGETT**

I, David St.John Raggett, declare and state as follows:

1. Except as otherwise indicated, I make this declaration based upon my personal knowledge.

2. Attached as Exhibit A is a copy of a document entitled *HTML+ (Hypertext markup language)* dated July 23, 1993 [hereinafter "*Raggett I*"] that I authored and first posted to the public WWW-Talk Internet discussion group on July 23, 1993. I have reviewed Exhibit A and it is a true and accurate copy of *Raggett I* as I posted it on July 23, 1993.

3. *Raggett I* is the second draft version of the HTML+ specification that I made publicly available. I am aware that a large number of postings from other people involved in the creation of HTML standards were made to the WWW-Talk discussion group that commented on the contents of *Raggett I* and other drafts of the HTML+ specification.

4. Attached as Exhibit B is a copy of a posting I made to the WWW-Talk discussion group on June 14, 1993 entitled *HTML+ support for eqn & Postscript* [hereinafter "*Raggett II*"] replying to comments made by others and posted on the WWW-Talk discussion group. I have reviewed Exhibit B and it is a true and accurate copy of *Raggett II* as I posted it on June 14, 1993.

5. All WWW-Talk discussion group postings, including *Raggett II* and the posting with *Raggett I*, were contemporaneously posted on the Internet and were made publicly available. To my knowledge, all WWW-Talk discussion group postings, including *Raggett II* and the posting with *Raggett I*, have remained publicly available on the Internet since the date they were posted.

6. A true and correct copy of the first posting with *Raggett I* is currently available at: http://ksi.cpsc.ucalgary.ca/archives/WWW-TALK/www-talk-1993q3.messages/282.html. A true and correct copy of *Raggett I* is currently available at: http://citeseer.nj.nec.com/raggett93html.html. A true and correct copy of *Raggett II* is currently available at: http://ksi.cpsc.ucalgary.ca/archives/ WWW-TALK/www-talk-1993q2.messages/467.html.

**********

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

David Raggett — 4th October 2003

David Raggett — Date

# Exhibit A

[Included as Attachment A above]

# Exhibit B

[Included as Attachment B above]

D

+0207 250 3443 15:44 25-06-04

Attachment D

U.S. Patent No. 5,838,906

Patent Application

US005838906A

# United States Patent [19]

Doyle et al.

[11] Patent Number: 5,838,906

[45] Date of Patent: Nov. 17, 1998

[54] **DISTRIBUTED HYPERMEDIA METHOD FOR AUTOMATICALLY INVOKING EXTERNAL APPLICATION PROVIDING INTERACTION AND DISPLAY OF EMBEDDED OBJECTS WITHIN A HYPERMEDIA DOCUMENT**

[75] Inventors: **Michael D. Doyle**, Alameda; **David C. Martin**, San Jose; **Cheong S. Ang**, Pacifica, all of Calif.

[73] Assignee: **The Regents of the University of California**, Oakland, Calif.

[21] Appl. No.: **324,443**

[22] Filed: **Oct. 17, 1994**

[51] **Int. Cl.**[6] ..... **C06F 9/44**; C06F 15/16; C06F 17/30

[52] **U.S. Cl.** ..... **395/200.32**; 395/200.28; 395/680; 395/685; 345/326; 345/346; 707/501; 707/513; 707/515; 707/516

[58] **Field of Search** ..... 395/157, 200.03, 395/161, 118, 144, 145, 146, 147, 148, 683, 777, 778, 762, 326, 333, 334, 335, 676, 682, 685, 684, 200.32, 200.33, 200.47–200.49; 707/501, 513, 515, 516; 345/326, 343, 346

[56] **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 4,815,029 | 3/1989 | Barker et al. | 707/516 |
| 4,847,604 | 7/1989 | Doyle | 340/706 |
| 4,949,248 | 8/1990 | Caro | 395/200.03 |
| 5,146,553 | 9/1992 | Noguchi et al. | 707/516 |
| 5,202,828 | 4/1993 | Vertelney et al. | 364/419 |
| 5,204,947 | 4/1993 | Bernstein et al. | 395/157 |
| 5,206,951 | 4/1993 | Khoyi et al. | 395/683 |
| 5,274,821 | 12/1993 | Rouquie | 395/705 |
| 5,307,499 | 4/1994 | Yin | 395/700 |
| 5,321,806 | 6/1994 | Meinerth et al. | 395/162 |
| 5,321,808 | 6/1994 | Rupp et al. | 395/164 |
| 5,347,632 | 9/1994 | Filepp et al. | 395/200.09 |

(List continued on next page.)

OTHER PUBLICATIONS

Stephen Le Hunte, "<EEMBED>—Embedded Objects", HTML Reference Library—HTMLIB v2.1, 1995: n.pag. Online. Internet.

"A Little History of the world Wide Web", n.pag. Online. Internet: available http://www.w3.org/History.html.

"NCSA Mosaic Version Information", n.pag. Online. Internet: available http://www.ncsa.uiuc.edu/SDG/Software.

"The second phase of the revolution", WIRED, Oct. 1994, pp. 116–152.

(List continued on next page.)

*Primary Examiner*—Dinh C. Dung

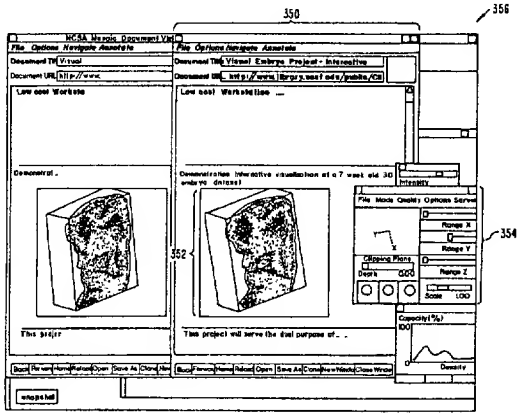
*Attorney, Agent, or Firm*—Townsend and Townsend and Crew LLP

[57] **ABSTRACT**

A system allowing a user of a browser program on a computer connected to an open distributed hypermedia system to access and execute an embedded program object. The program object is embedded into a hypermedia document much like data objects. The user may select the program object from the screen. Once selected the program object executes on the user's (client) computer or may execute on a remote server or additional remote computers in a distributed processing arrangement. After launching the program object, the user is able to interact with the object as the invention provides for ongoing interprocess communication between the application object (program) and the browser program. One application of the embedded program object allows a user to view large and complex multi-dimensional objects from within the browser's window. The user can manipulate a control panel to change the viewpoint used to view the image. The invention allows a program to execute on a remote server or other computers to calculate the viewing transformations and send frame data to the client computer thus providing the user of the client computer with interactive features and allowing the user to have access to greater computing power than may be available at the user's client computer.

**10 Claims, 9 Drawing Sheets**

Microfiche Appendix Included
(4 Microfiche, 375 Pages)



U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 5,367,635 | 11/1994 | Bauer et al. | 395/200.32 |
| 5,390,314 | 2/1995 | Swanson | 395/500 |
| 5,418,908 | 5/1995 | Keller et al. | 395/200.32 |
| 5,544,320 | 8/1996 | Konrad | 395/200.09 |
| 5,581,686 | 12/1996 | Koppolu et al. | 395/340 |
| 5,606,493 | 2/1997 | Duscher et al. | 395/200.32 |
| 5,652,876 | 7/1997 | Ashe et al. | 707/516 |

OTHER PUBLICATIONS

Vetter, Ronald "Mosaic and the World–Wide Web," Computer Magazine, v.27, Iss.10, pp. 49–57, Oct. 1994.

Wynne et al. "Lean Management, Group Support Systems, and Hypermedia: a Combination Whose Time Has Come," System Sciences, 1993 Anuall Hawaii Int'l Conf., pp. 112–121.

Hansen, Wilfred "Andrew as a Multiparadigm Environment for Visual Languages," Visual Languages, 1993 IEEE Symposium, pp. 256–260.

Moran, Patrick "Tele–Nicer–slicer–Dicer: A New Tool for the Visualization of Large Volumetric Data", NCSA Technical Report (TRO14), Aug. 1993.

Berners–Lee "Hypertext Markup Language (HTML)", HTML Internet Draft, IIIR working Group, Jun. 1993.

University of Southern California's Mercury Project—"USC Mercury Project:Interface", Project Milestones, USC Press Release—obtained from Internet, http://www.usc.edu/dept/raiders/.

Hansen, Wilfred "Enhancing documents with embedded programs: How Ness extends in the Andrew ToolKit", IEEE Computer Language, 1990 International Conference.

Tani, M., et al., "Object–Oriented Video: Interaction with Real–World Objects Through Live Video", May 1992, p. 593–598.

Crowley, T., et al., "MMConf: An Infrastructure for Building Shared Multimedia Applications", CSCW 90 Proceedings, Oct. 1990, p. 329–342.

Davis, H., et al., "Towards An Integrated Information Environment With Open Hypermedia System", ACM ECHT Conference, Dec. 1992, pp. 181–190.

Ferrara, F., "The KIM Query System", Abstract, SIGCHI Bulletin, vol. 6, No. 3, Jul. 1994, pp. 30–39.

Gibbs, S., "Composite Multimedia and Active Objects", OOPSLA '91, pp. 97–112.

Davis, H., et al., "Microcosm: An Open Hypermedia System", Interchi '93, Apr. 1993, p. 526.

Vaziri, A., "Scientific Visualization in High–Speed Network Environments", Computer Networks and ISDN Systems 22, 1991, pp. 111–129.

Cullen, J., et al., "The Use of FTAM to access graphical pictures across wide area networks", Computer Networks and ISDN Systems, 1992, pp. 337–383.

Lashkari, Y.Z., et al., "PLX: A Proposal to Implement a General Broadcasting Facility in a Distributed Environment Running X Windows", Comput. & Graphics, vol. 16, No. 2, pp. 143–149, 1992.

Kirste, T., "Spacepicture—An Interactive Hypermedia Satellite Image Archival System", Comput. & Graphics, vol. 17, No. 3, pp. 251–260, 1993.

Coulson, G., et al., "Extensions to ANSA for Multimedia Computing", Computers Networks and ISDN Systems 25, 1992, pp. 305–323.

Huynh, Duong Le, et al., "PIX: An Object–Oriented Network Graphics Environment", Comput. & Graphics, vol. 17, No. 3, pp. 295–304, 1993.

Berners–Lee, T.J., et al., The World–Wide Web, Computer Networks and ISDN Systems 25, 1992, pp. 454–459.

Shackelford, D.E., et al., "The Architecture and Implementation of a Distributed Hypermedia Storage System", Hypertext '93 Proceedings, Nov. 1993, pp. 1–13.

Labriola, D., "Remote Possibilities", PC Magazine, Jun. 14, 1994, pp. 223–228.

Udell, J., "Visual Basic Custom Controls Meet OLE", Byte Magazine, Mar. 1994, pp. 197–200.

Sarna, D.E., et al., "OLE Gains Without (Much) Pain", Datamation Magazine, Jun. 15, 1994, pp. 31 and 113.

Rizzo, J., "What's OpenDoc?", MacUser magazine, Apr. 1994, pp. 119–123.

Fogarty, K., et al., "Microsoft's OLE can be network Trojan Horse", Network World Magazine, Jun. 27, 1994, vol. 11, No. 26, pp. 1 and 75.

"Cello WWW Browser Release 1.01a", Article obtained from the Internet, ftp.law.cornell.edu/pub/L11/Cello no DDE, Mar. 16, 1994, pp. 2–9.

"OLE 2.0: Death to Monoliths", Byte Magazine, Mar. 1994, p. 122.

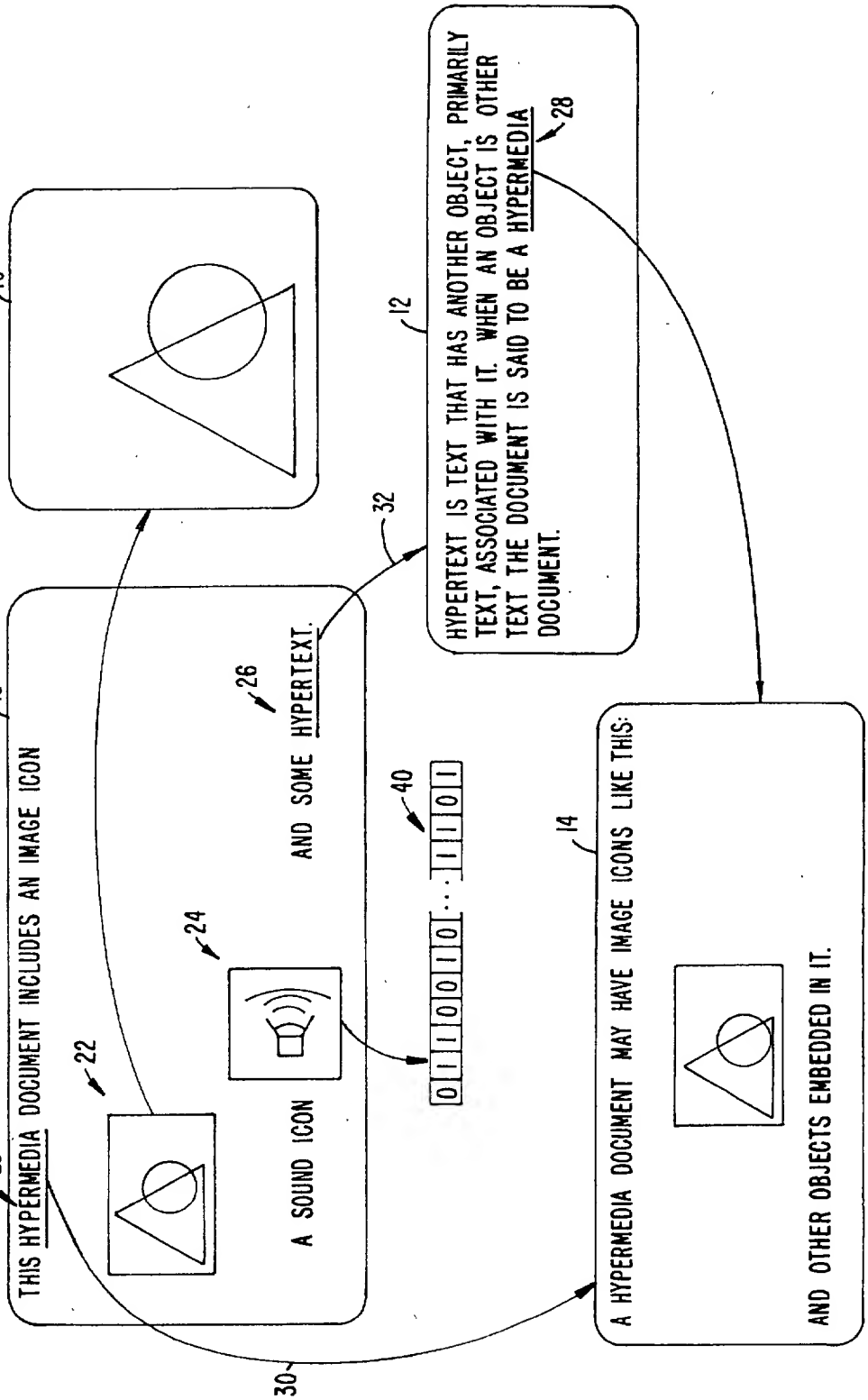



FIG. 1. PRIOR ART

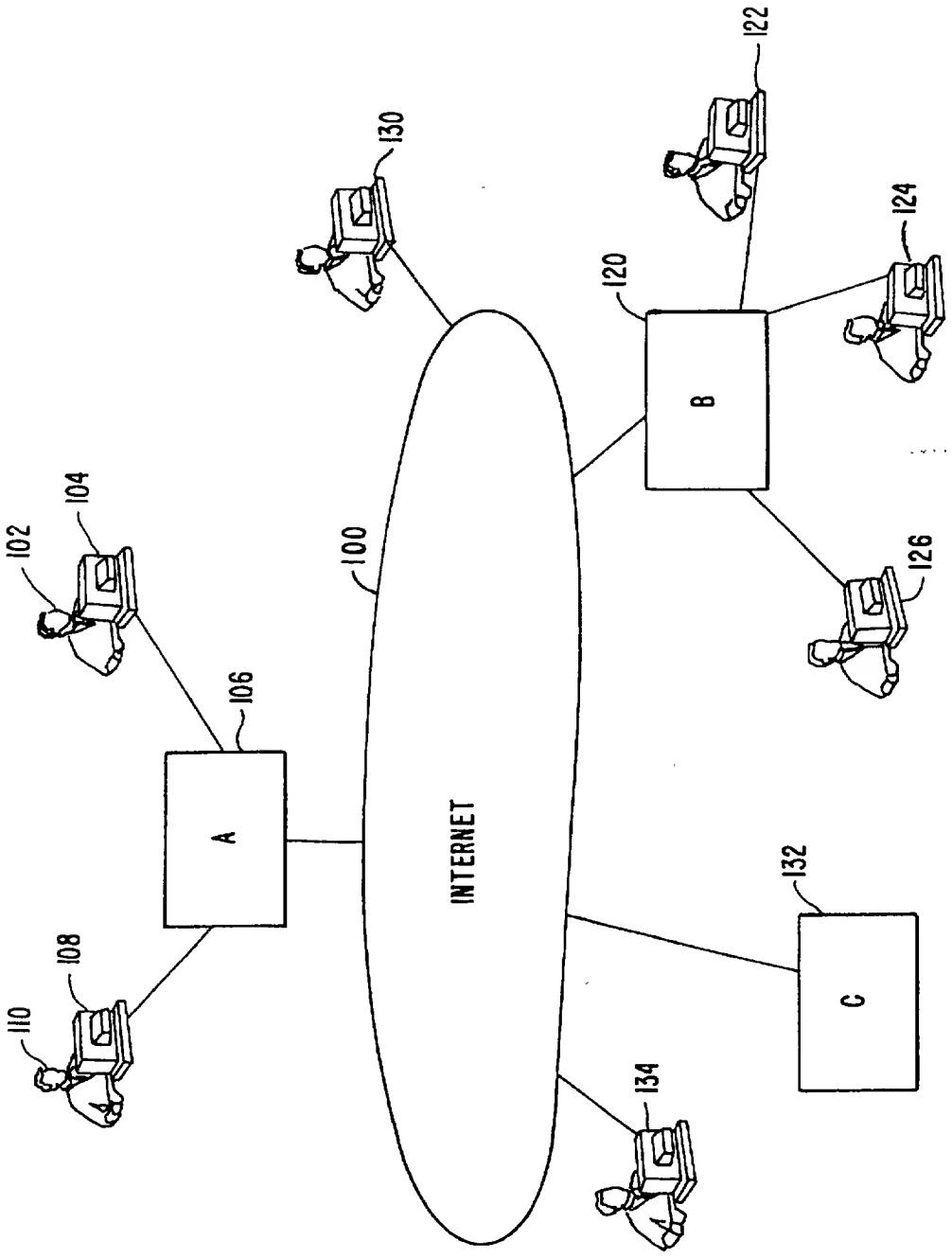



FIG. 2. PRIOR ART

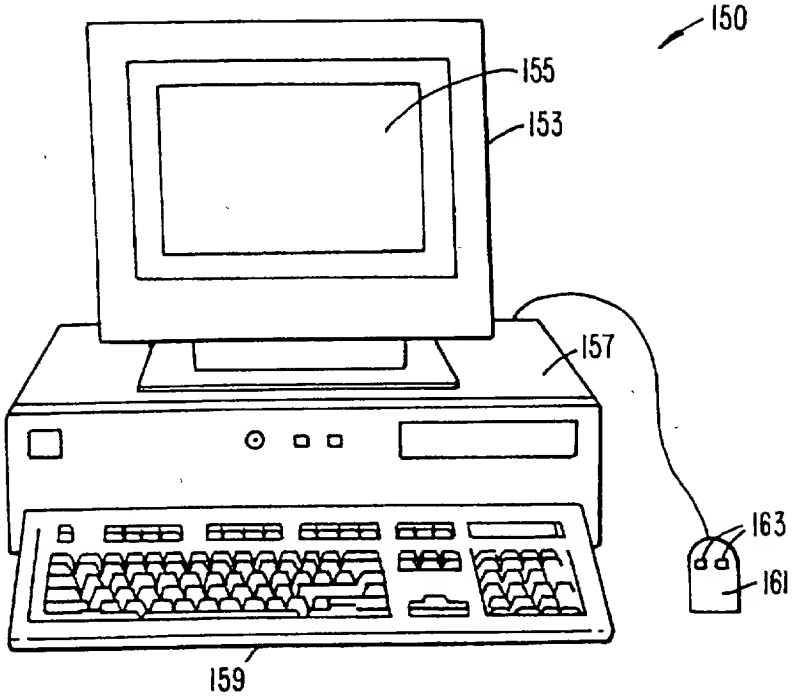

150
155
153
157
163
161
159

FIG. 3.

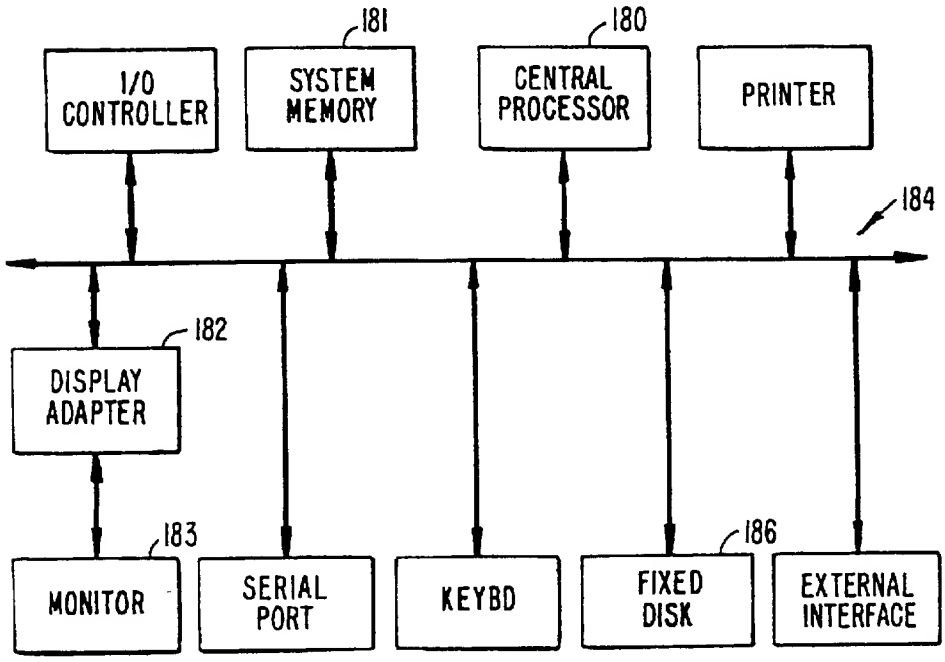

181
180
I/O CONTROLLER
SYSTEM MEMORY
CENTRAL PROCESSOR
PRINTER
184
182
DISPLAY ADAPTER
183
186
MONITOR
SERIAL PORT
KEYBD
FIXED DISK
EXTERNAL INTERFACE

FIG. 4.

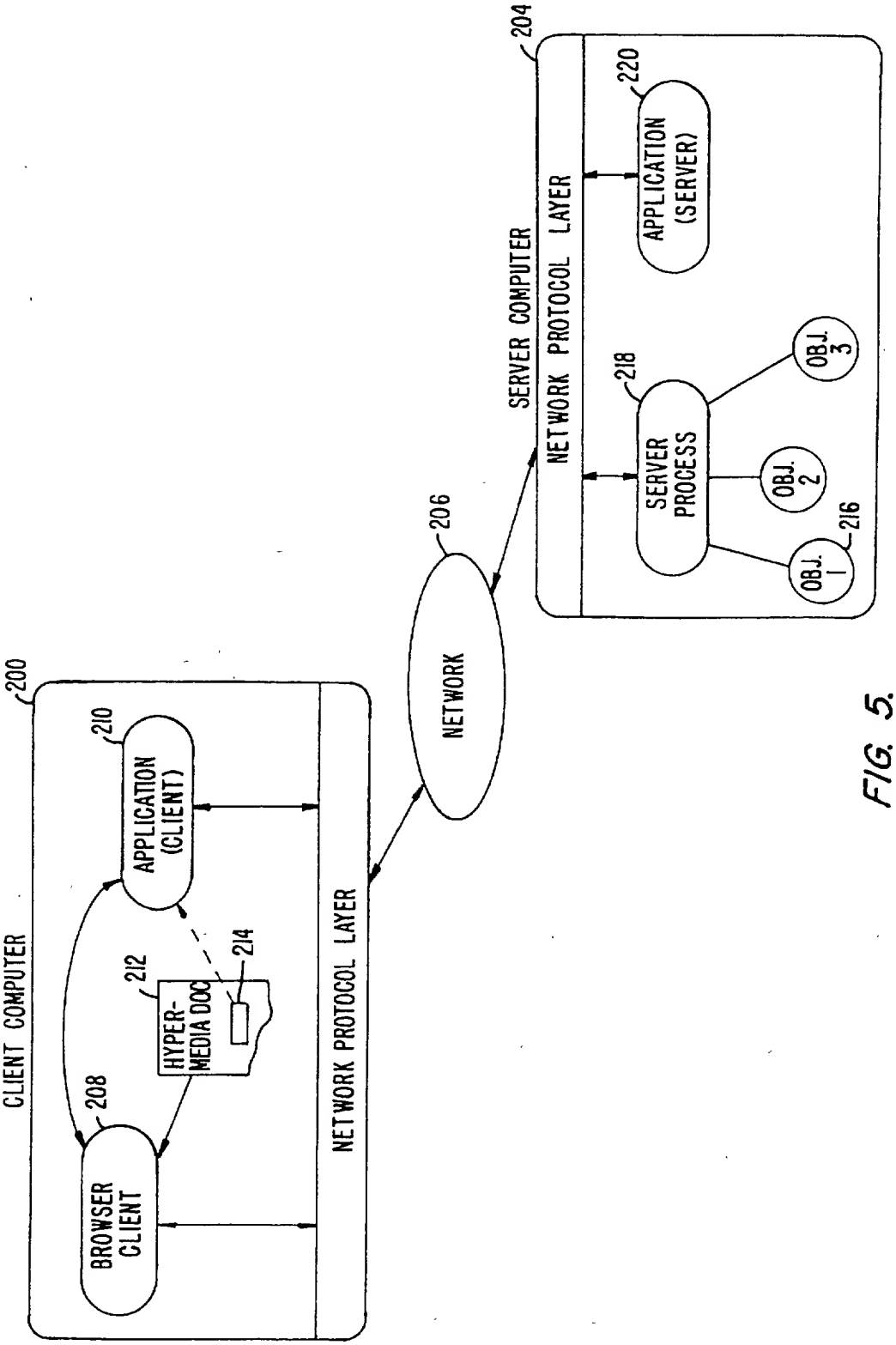

200
CLIENT COMPUTER
210
APPLICATION (CLIENT)
212
HYPER-MEDIA DOC
214
208
BROWSER CLIENT
NETWORK PROTOCOL LAYER
206
NETWORK
204
SERVER COMPUTER
NETWORK PROTOCOL LAYER
220
APPLICATION (SERVER)
218
SERVER PROCESS
OBJ. 1
OBJ. 2
OBJ. 3
216

FIG. 5.

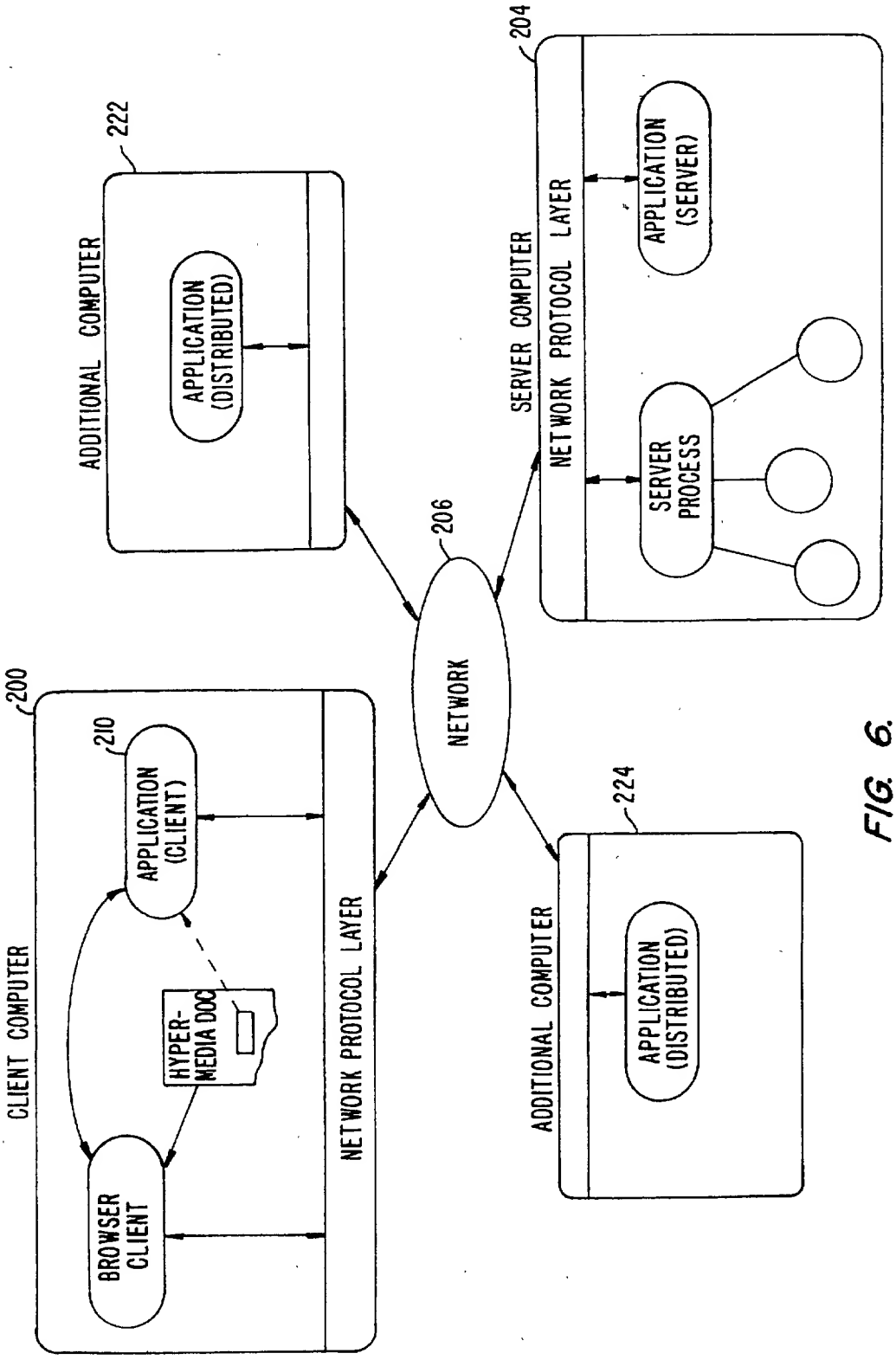

200
CLIENT COMPUTER
210
APPLICATION (CLIENT)
HYPER-MEDIA DOC
BROWSER CLIENT
NETWORK PROTOCOL LAYER
206
NETWORK
222
ADDITIONAL COMPUTER
APPLICATION (DISTRIBUTED)
204
SERVER COMPUTER
NETWORK PROTOCOL LAYER
APPLICATION (SERVER)
SERVER PROCESS
224
ADDITIONAL COMPUTER
APPLICATION (DISTRIBUTED)

FIG. 6.

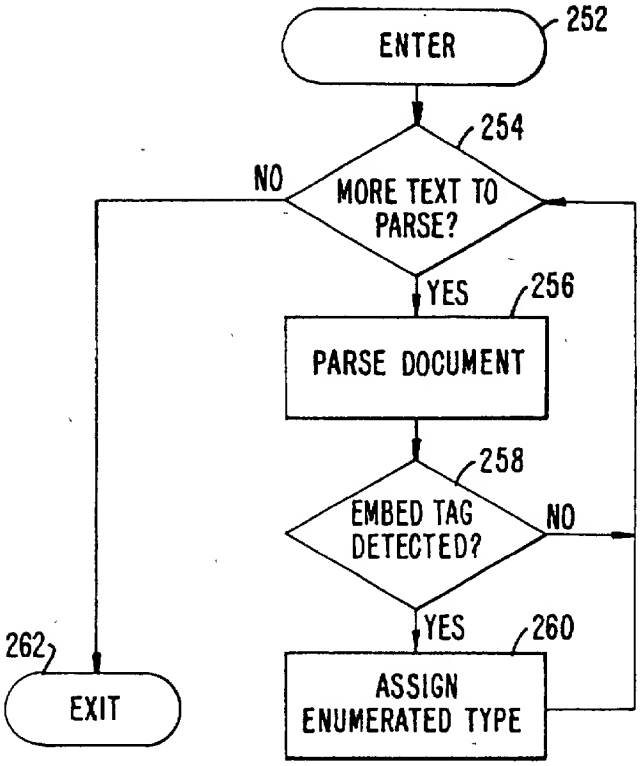

ENTER
252
254
MORE TEXT TO PARSE?
NO
YES
256
PARSE DOCUMENT
258
EMBED TAG DETECTED?
NO
YES
260
ASSIGN ENUMERATED TYPE
262
EXIT

FIG. 7A.

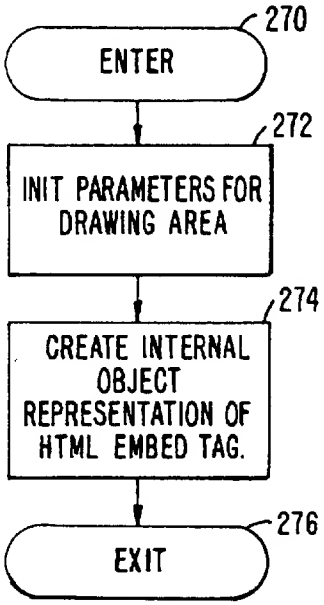

ENTER
270
INIT PARAMETERS FOR DRAWING AREA
272
CREATE INTERNAL OBJECT REPRESENTATION OF HTML EMBED TAG.
274
EXIT
276

FIG. 7B.

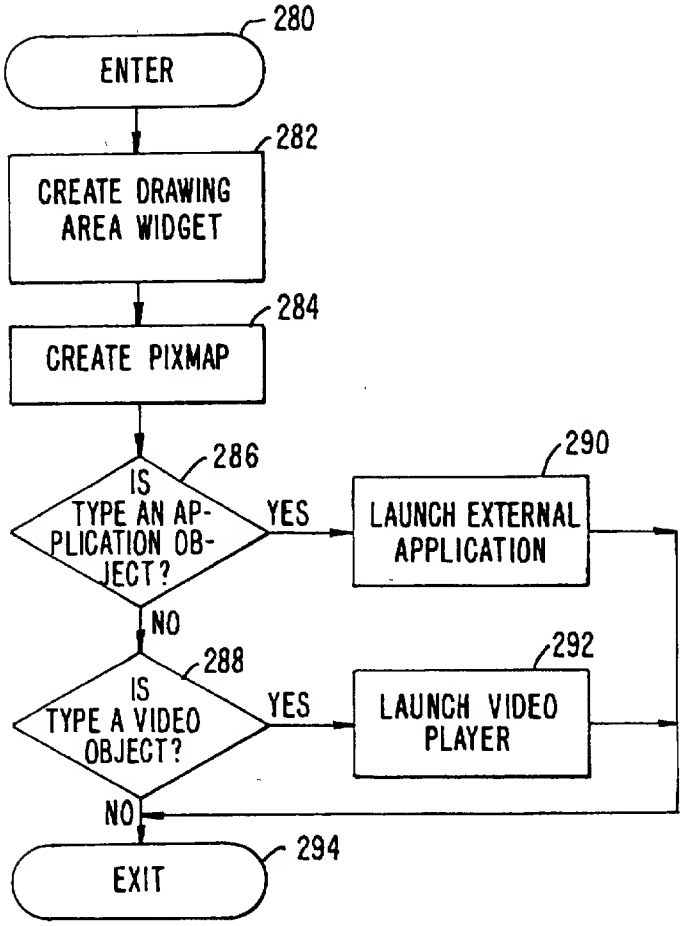

280
ENTER
282
CREATE DRAWING AREA WIDGET
284
CREATE PIXMAP
286
IS TYPE AN APPLICATION OBJECT?
YES
290
LAUNCH EXTERNAL APPLICATION
NO
288
IS TYPE A VIDEO OBJECT?
YES
292
LAUNCH VIDEO PLAYER
NO
294
EXIT

FIG. 8A.

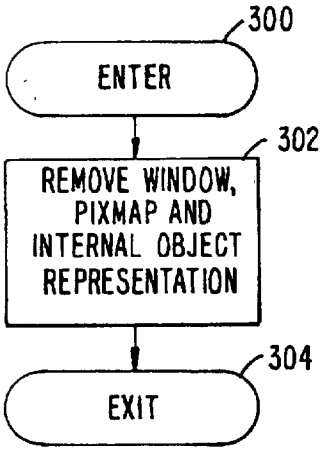

300
ENTER
302
REMOVE WINDOW, PIXMAP AND INTERNAL OBJECT REPRESENTATION
304
EXIT

FIG. 8B.

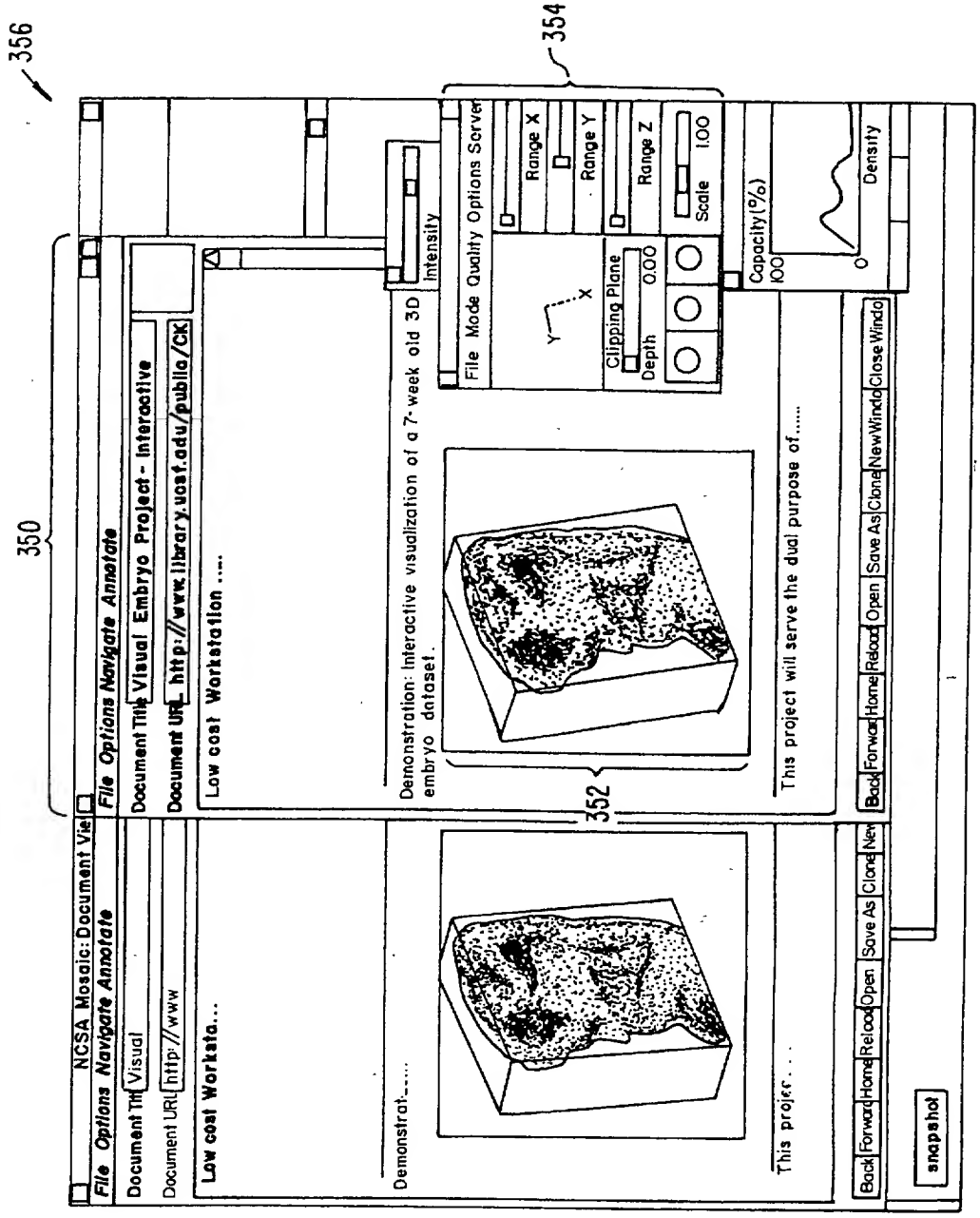

356
354
350
352
NCSA Mosaic: Document Vie
File Options Navigate Annotate
Document Title: Visual
Document URL: http://www
Low cost Worksta...
Demonstrat.....
This projer...
Back Forward Home Reload Open Save As Clone New
snapshot
File Options Navigate Annotate
Document Title: Visual Embryo Project - Interactive
Document URL: http://www.library.uoat.adu/public/CK
Low cost Workstation ....
Demonstration: Interactive visualization of a 7-week old 3D embryo dataset.
This project will serve the dual purpose of.....
Intensity
File Mode Quality Options Server
Range X
Range Y
Range Z
Scale 1.00
Clipping Plane
Depth 0.00
Capacity(%)
100
0
Density
Back Forwar Home Reload Open Save As Clone NewWindo Close Windo

FIG. 9.

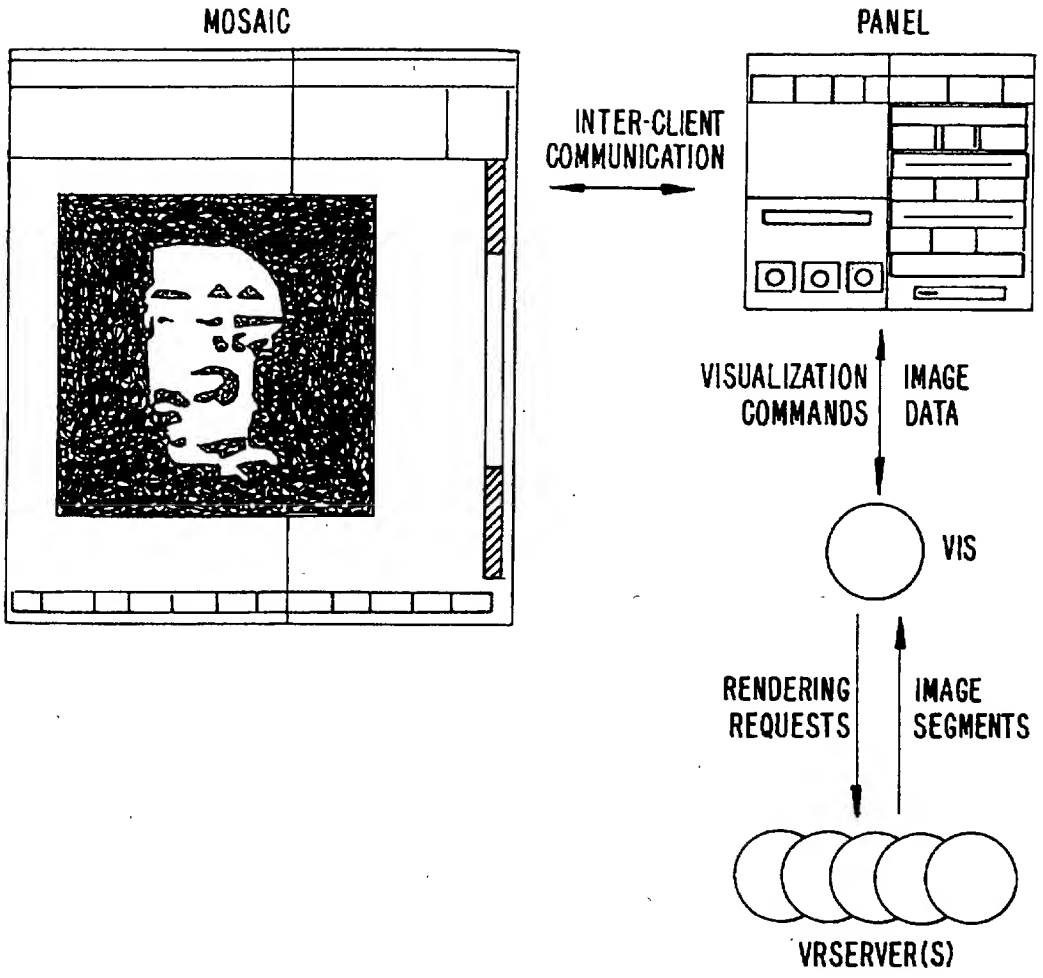

MOSAIC
PANEL
INTER-CLIENT COMMUNICATION
VISUALIZATION COMMANDS
IMAGE DATA
VIS
RENDERING REQUESTS
IMAGE SEGMENTS
VRSERVER(S)

FIG. 10.

# DISTRIBUTED HYPERMEDIA METHOD FOR AUTOMATICALLY INVOKING EXTERNAL APPLICATION PROVIDING INTERACTION AND DISPLAY OF EMBEDDED OBJECTS WITHIN A HYPERMEDIA DOCUMENT

## NOTICE REGARDING COPYRIGHTED MATERIAL

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office file or records, but otherwise reserves all copyright rights whatsoever.

## BACKGROUND OF THE INVENTION

This invention relates generally to manipulating data in a computer network, and specifically to retrieving, presenting and manipulating embedded program objects in distributed hypermedia systems.

Computer networks are becoming increasingly popular as a medium for locating and accessing a wide range of data from locations all over the world. The most popular global network is the Internet with millions of computer systems connected to it. The Internet has become popular due to widely adopted standard protocols that allow a vast interconnection of computers and localized computer networks to communicate with each other. Computer systems connected to a network such as the Internet may be of varying types, e.g., mainframes, workstations, personal computers, etc. The computers are manufactured by different companies using proprietary hardware and operating systems and thus have incompatibilities in their instruction sets, busses, software, file formats and other aspects of their architecture and operating systems. Localized computer networks connected to the Internet may be incompatible with other computer systems and localized networks in terms of the physical layer of communication including the specific hardware used to implement the network. Also, different networks use differing, incompatible protocols for transferring information and are not able to communicate with each other without a translation mechanism such as a "gateway".

The Internet provides a uniform and open standard for allowing various computers and networks to communicate with each other. For example, the Internet uses Transfer Control Protocol/Internet Protocol ("TCP/IP") that defines a uniform packet-switched communication standard which is ultimately used in every transfer of information that takes place over the Internet.

Other Internet standards are the HyperText Transmission Protocol ("HTTP") that allows hypertext documents to be exchanged freely among any computers connected to the Internet and HyperText Markup Language ("HTML") that defines the way in which hypertext documents designate links to information. See, e.g., Berners-Lee, T. J., "The world-wide web," Computer Networks and ISDN Systems 25 (1992).

A hypertext document is a document that allows a user to view a text document displayed on a display device connected to the user's computer and to access, retrieve and view other data objects that are linked to hypertext words or phrases in the hypertext document. In a hypertext document, the user may "click on," or select, certain words or phrases in the text that specify a link to other documents, or data objects. In this way, the user is able to navigate easily among data objects. The data objects may be local to the user's computer system or remotely located over a network. An early hypertext system is Hypercard, by Apple Computer, Inc. Hypercard is a standalone system where the data objects are local to the user's system.



When a user selects a phrase in a hypertext document that has an associated link to another document, the linked document is retrieved and displayed on the user's display screen. This allows the user to obtain more information in an efficient and easy manner. This provides the user with a simple, intuitive and powerful way to "branch off" from a main document to learn more about topics of interest.

Objects may be text, images, sound files, video data, documents or other types of information that is presentable to a user of a computer system. When a document is primarily text and includes links to other data objects according to the hypertext format, the document is said to be a hypertext document. When graphics, sound, video or other media capable of being manipulated and presented in a computer system is used as the object linked to, the document is said to be a hypermedia document. A hypermedia document is similar to a hypertext document, except that the user is able to click on images, sound icons, video icons, etc., that link to other objects of various media types, such as additional graphics, sound, video, text, or hypermedia or hypertext documents.

FIG. **1** shows examples of hypertext and hypermedia documents and links associating data objects in the documents to other data objects. Hypermedia document **10** includes hypertext **20**, an image icon at **22**, a sound icon at **24** and more hypertext **26**. FIG. **1** shows hypermedia document **10** substantially as it would appear on a user's display screen. The user is able to select, or "click" on icons and text on a display screen by using an input device, such as a mouse, in a manner well-known in the art.

When the user clicks on the phrase "hypermedia," software running on the user's computer obtains the link associated with the phrase, symbolically shown by arrow **30**, to access hypermedia document **14**. Hypermedia document **14** is retrieved and displayed on the user's display screen. Thus, the user is presented with more information on the phrase "hypermedia." The mechanism for specifying and locating a linked object such as hypermedia document **14** is an HTML "element" that includes an object address in the format of a Uniform Resource Locator (URL).

Similarly, additional hypertext **26** can be selected by the user to access hypertext document **12** via link **32** as shown in FIG. **1**. If the user selects additional hypertext **26**, then the text for hypertext document **12** is displayed on the user screen. Note that hypertext document **12**, itself, has hypertext at **28** Thus, the user can click on the phrase "hypermedia" while viewing document **12** to access hypermedia document **14** in a manner similar to that discussed above.

Documents, and other data objects, can be referenced by many links from many different source documents. FIG. **1** shows document **14** serving as a target link for both documents **10** and **12**. A distributed hypertext or hypermedia document typically has many links within it that specify many different data objects located in computers at different geographical locations connected by a network. Hypermedia document **10** includes image icon **22** with a link to image **16**. One method of viewing images is to include an icon, or other indicator, within the text.

Typically, the indicator is a very small image and may be a scaled down version of the full image. The indicator may

be shown embedded within the text when the text is displayed on the display screen. The user may select the indicator to obtain the full image. When the user clicks on image icon **22** browser software executing on the user's computer system retrieves the corresponding full image, e.g., a bit map, and displays it by using external software called a "viewer." This results in the full image, represented by image **16**, being displayed on the screen.

An example of a browser program is the National Center for Supercomputing Application's (NCSA) Mosaic software developed by the University of Illinois at Urbana/Champaign, Ill. Another example is "Cello" available on the Internet at http://www.law.cornell.edu/. Many viewers exist that handle various file formats such as ".TIF," ".GIF," formats. When a browser program invokes a viewer program, the viewer is launched as a separate process. The view displays the full image in a separate "window" (in a windowing environment) or on a separate screen. This means that the browser program is no longer active while the viewer is active. By using indicators to act as place holders for full images that are retrieved and displayed only when a user selects the indicator, data traffic over the network is reduced. Also, since the retrieval and display of large images may require several seconds or more of transfer time the user does not have to wait to have images transferred that are of no interest to the user.

Returning to FIG. 1, another type of data object is a sound object shown as sound icon **24** within the hypermedia document. When the user selects sound icon **24**, the user's computer accesses sound data shown symbolically by data file **40**. The accessed sound data plays through a speaker or other audio device.

As discussed above, hypermedia documents allow a user to access different data objects. The objects may be text, images, sound files, video, additional documents, etc. As used in this specification, a data object is information capable of being retrieved and presented to a user of a computer system. Some data objects include executable code combined with data. An example of such a combination is a "self-extracting" data object that includes code to "unpack" or decompress data that has been compressed to make it smaller before transferring. When a browser retrieves an object such as a self-extracting data object the browser may allow the user to "launch" the self-extracting data object to automatically execute the unpacking instructions to expand the data object to its original size. Such a combination of executable code and data is limited in that the user can do no more than invoke the code to perform a singular function such as performing the self-extraction after which time the object is a standard data object.

Other existing approaches to embedding interactive program objects in documents include the Object Linking and Embedding (OLE) facility in Microsoft Windows, by Microsoft Corp., and OpenDoc, by Apple Computer, Inc. At least one shortcoming of these approaches is that neither is capable of allowing a user to access embedded interactive program objects in distributed hypermedia documents over networks.

FIG. **2** is an example of a computer network. In FIG. **2**, computer systems are connected to Internet **100**, although in practice Internet **100** may be replaced by any suitable computer network. In FIG. **2**, a user **102** operates a small computer **104**, such as a personal computer or a work station. The user's computer is equipped with various components, such as user input devices (mouse, trackball, keyboard, etc.), a display device (monitor, liquid crystal display (LCD), etc.), local storage (hard disk drive, etc.), and other components. Typically, small computer **104** is connected to a larger computer, such as server A at **106**. The larger computer may have additional users and computer systems connected to it, such as computer **108** operated by user **110**. Any group of computers may form a localized network. A localized network does not necessarily adopt the uniform protocols of the larger interconnecting network (i.e., Internet **100**) and is more geographically constrained than the larger network. The localized network may connect to the larger network through a "gateway" or "node" implemented on, for example, a server.

Internet **100** connects other localized networks, such as server B at **120**, which interconnects users **122**, **124** and **126** and their respective computer systems to Internet **100**. Internet **100** is made up of many interconnected computer systems and communication links. Communication links may be by hardwire, fiber optic cable, satellite or other radio wave propagation, etc. Data may move from server A to server B through any number of intermediate servers and communication links or other computers and data processing equipment not shown in FIG. **2** but symbolically represented by Internet **100**.

A user at a workstation or personal computer need not connect to the Internet via a larger computer, such as server A or server B. This is shown, for example, by small computer **130** connected directly to Internet **100** as by a telephone modem or other link. Also, a server need not have users connected to it locally, as is shown by server C at **132** of FIG. **2**. Many configurations of large and small computers are possible.

Typically, a computer on the Internet is characterized as either a "client" or "server" depending on the role that the computer is playing with respect to requesting information or providing information. Client computers are computers that typically request information from a server computer which provides the information. For this reason, servers are usually larger and faster machines that have access to many data files, programs, etc., in a large storage associated with the server. However, the role of a server may also be adopted by a smaller machine depending on the transaction. That is, user **110** may request information via their computer **108** from server A. At a later time, server A may make a request for information from computer **108**. In the first case, where computer **108** issues a request for information from server A, computer **108** is a "client" making a request of information from server A. Server A may have the information in a storage device that is local to Server A or server A may have to make requests of other computer systems to obtain the information. User **110** may also request information via their computer **108** from a server, such as server B located at a remote geographical location on the Internet. However, user **110** may also request information from a computer, such as small computer **124**, thus placing small computer **124** in the role of a "server." For purposes of this specification, client and server computers are categorized in terms of their predominant role as either an information requestor or provider. Clients are generally information requestors, while servers are generally information providers.

Referring again to FIG. **1**, data objects such as distributed hypermedia documents **10**, **12** and **14**, image **16** and sound data file **40**, may be located at any of the computers shown in FIG. **2**. Since these data objects may be linked to a document located on another computer the Internet allows for remote object linking.

For example, hypertext document **10** of FIG. **1** may be located at user **110**'s client computer **108**. When user **110**

makes a request by, for example, clicking on hypertext **20** (i.e., the phrase "hypermedia"), user **110**'s small client computer **108** processes links within hypertext document **10** to retrieve document **14**. In this example, we assume that document **14** is stored at a remote location on server B. Thus, in this example, computer **108** issues a command that includes the address of document **14**. This command is routed through server A and Internet **100** and eventually is received by server B. Server B processes the command and locates document **14** on its local storage. Server **14** then transfers a copy of the document back to client **108** via Internet **100** and server A. After client computer **108** receives document **14**, it is displayed so that user **110** may view it.

Similarly, image object **16** and sound data file **40** may reside at any of the computers shown in FIG. **2**. Assuming image object **16** resides on server C when user **110** clicks on image icon **22**, client computer **108** generates a command to retrieve image object **16** to server C. Server C receives the command and transfers a copy of image object **16** to client computer **108**. Alternatively, an object, such as sound data file **40**, may reside on server A so that it is not necessary to traverse long distances via the Internet in order to retrieve the data object.

The Internet is said to provide an "open distributed hypermedia system." It is an "open" system since Internet **100** implements a standard protocol that each of the connecting computer systems, **106, 130, 120, 132** and **134** must implement (TCP/IP). It is a "hypermedia" system because it is able to handle hypermedia documents as described above via standards such as the HTTP and HTML hypertext transmission and mark up standards, respectively. Further, it is a "distributed" system because data objects that are imbedded within a document may be located on many of the computer systems connected to the Internet. An example of an open distributed hypermedia system is the so-called "world-wide web" implemented on the Internet and discussed in papers such as the Berners-Lee reference given above.

The open distributed hypermedia system provided by the Internet allows users to easily access and retrieve different data objects located in remote geographic locations on the Internet. However, this open distributed hypermedia system as it currently exists has shortcomings in that today's large data objects are limited largely by bandwidth constraints in the various communication links in the Internet and localized networks, and by the limited processing power, or computing constraints, of small computer systems normally provided to most users. Large data objects are difficult to update at frame rates fast enough (e.g., 30 frames per second) to achieve smooth animation. Moreover, the processing power needed to perform the calculations to animate such images in real time does not exist on most workstations, not to mention personal computers. Today's browsers and viewers are not capable of performing the computation necessary to generate and render new views of these large data objects in real time.

For example, the Internet's open-distributed hypermedia system allows users to view still images. These images are simple non-interactive two-dimensional images, similar to photographs. Much digital data available today exists in the form of high-resolution multi-dimensional image data (e.g., three dimensional images) which is viewed on a computer while allowing the user to perform real time viewing transformations on the data in order for the user to better understand the data.

An example of such type of data is in medical imaging where advanced scanning devices, such as Magnetic Resonance Imaging (MRI) and Computed Tomography (CT), are widely used in the fields of medicine, quality assurance and meteorology to present physicians, technicians and meteorologists with large amounts of data in an efficient way. Because visualization of the data is the best way for a user to grasp the data's meaning, a variety of visualization techniques and real time computer graphics methods have been developed. However, these systems are bandwidth-intensive and compute-intensive and often require multiprocessor arrays and other specialized graphics hardware to carry them out in real time. Also, large amounts of secondary storage for data are required. The expense of these requirements has limited the ability of researchers to readily exchange findings since these larger computers required to store, present and manipulate images are not available to many of the researchers that need to have access to the data.

On the other hand, small client computers in the form of personal computers or workstations such as client computer **108** of FIG. **2** are generally available to a much larger number of researchers. Further, it is common for these smaller computers to be connected to the Internet. Thus, it is desirable to have a system that allows the accessing, display and manipulation of large amounts of data, especially image data, over the Internet to a small, and relatively cheap, client computer.

Due to the relatively low bandwidth of the Internet (as compared to today's large data objects) and the relatively small amount of processing power available at client computers, many valuable tasks performed by computers cannot be performed by users at client computers on the Internet. Also, while the present open distributed hypermedia system on the Internet allows users to locate and retrieve data objects it allows users very little, if any, interaction with these data objects. Users are limited to traditional hypertext and hypermedia forms of selecting linked data objects for retrieval and launching viewers or other forms of external software to have the data objects presented in a comprehensible way.

Thus, it is desirable to have a system that allows a user at a small client computer connected to the Internet to locate, retrieve and manipulate data objects when the data objects are bandwidth-intensive and compute-intensive. Further, it is desirable to allow a user to manipulate data objects in an interactive way to provide the user with a better understanding of information presented and to allow the user to accomplish a wider variety of tasks.

## SUMMARY OF THE INVENTION

The present invention provides a method for running embedded program objects in a computer network environment. The method includes the steps of providing at least one client workstation and one network server coupled to the network environment where the network environment is a distributed hypermedia environment; displaying, on the client workstation, a portion of a hypermedia document received over the network from the server, where the hypermedia document includes an embedded controllable application; and interactively controlling the embedded controllable application from the client workstation via communication sent over the distributed hypermedia environment.

The present invention allows a user at a client computer connected to a network to locate, retrieve and manipulate objects in an interactive way. The invention not only allows the user to use a hypermedia format to locate and retrieve program objects, but also allows the user to interact with an

application program located at a remote computer. Interprocess communication between the hypermedia browser and the embedded application program is ongoing after the program object has been launched. The user is able to use a vast amount of computing power beyond that which is contained in the user's client computer.

In one application, high resolution three dimensional images are processed in a distributed manner by several computers located remotely from the user's client computer. This amounts to providing parallel distributed processing for tasks such as volume rendering or three dimensional image transformation and display. Also, the user is able to rotate, scale and otherwise reposition the viewpoint with respect to these images without exiting the hypermedia browser software. The control and interaction of viewing the image may be provided within the same window that the browser is using assuming the environment is a "windowing" environment. The viewing transformation and volume rendering calculations may be performed by remote distributed computer systems.

Once an image representing a new viewpoint is computed the frame image is transmitted over the network to the user's client computer where it is displayed at a designated position within a hypermedia document. By transmitting only enough information to update the image, the need for a high bandwidth data connection is reduced. Compression can be used to further reduce the bandwidth requirements for data transmission.

Other applications of the invention are possible. For example, the user can operate a spreadsheet program that is being executed by one or more other computer systems connected via the network to the user's client computer. Once the spreadsheet program has calculated results, the results may be sent over the network to the user's client computer for display to the user. In this way, computer systems located remotely on the network can be used to provide the computing power that may be required for certain tasks and to reduce the data bandwidth by only transmitting results of the computations.

Still other applications of the present invention are possible, as disclosed in the specification, below.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates examples of hypertext and hypermedia documents and links;

FIG. **2** is an example of a computer network;

FIG. **3** is an illustration of a computer system suitable for use with the present invention;

FIG. **4** is an illustration of basic subsystems in the computer system of FIG. **3**;

FIG. **5** is an illustration of an embodiment of the invention using a client computer, server computer and a network;

FIG. **6** shows another embodiment of the present invention using additional computers on the network;

FIG. **7A** is a flowchart of some of the functionality within the HTMLparse.c file;

FIG. **7B** is a flowchart of some of the functionality within the HTMLformat.c file;

FIG. **8A** is a flowchart of some of the functionality within the HTMLwidget.c file;

FIG. **8B** is a flowchart of some of the functionality within the HTML.c file;

FIG. **9** is a screen display generated in accordance with the present invention; and

FIG. **10** is a diagram of the various processes and data paths in the present invention.

## DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

375 pages of Source code on 4 microfiche Appendices A and B are provided to this specification. The source code should be consulted to provide details of a specific embodiment of the invention in conjunction with the discussion of the routines in this specification. The source code in Appendix A includes NCSA Mosaic version 2.4 source code along with modifications to the source code to implement the present invention. Appendix B includes source code implementing an application program interface The source code is written in the "C" computer language to run on an X-Window platform.

FIG. **3** is an illustration of a computer system suitable for use with the present invention. FIG. **3** depicts but one example of many possible computer types or configurations capable of being used with the present invention. FIG. **3** shows computer system **150** including display device **153**, display screen **155**, cabinet **157**, keyboard **159** and mouse **161**. Mouse **161** and keyboard **159** are "user input devices." Other examples of user input devices are a touch screen, light pen, track ball, data glove, etc.

Mouse **161** may have one or more buttons such as buttons **163** shown in FIG. **3**. Cabinet **157** houses familiar computer components such as disk drives, a processor, storage means, etc. As used in this specification "storage means" includes any storage device used in connection with a computer system such as disk drives, magnetic tape, solid state memory, bubble memory, etc. Cabinet **157** may include additional hardware such as input/output (I/O) interface cards for connecting computer system **150** to external devices such as an optical character reader, external storage devices, other computers or additional devices.

FIG. **4** is an illustration of basic subsystems in computer system **150** of FIG. **3**. In FIG. **4**, subsystems are represented by blocks such as central processor **180**, system memory **181** consisting of random access memory (RAM) and/or read-only memory (ROM), display adapter **182**, monitor **183** (equivalent to display device **153** of FIG. **3**), etc. The subsystems are interconnected via a system bus **184**. Additional subsystems such as a printer, keyboard, fixed disk and others are shown. Peripherals and input/output (I/O) devices can be connected to the computer system by, for example serial port **185**. For example, serial port **185** can be used to connect the computer system to a modem for connection to a network or serial port **185** can be used to interface with a mouse input device. The interconnection via system bus **184** allows central processor **180** to communicate with each subsystem and to control the execution of instructions from system memory **181** or fixed disk **186**, and the exchange of information between subsystems. Other arrangements of subsystems and interconnections are possible.

FIG. **5** is an illustration of an embodiment of the invention using a client computer, server computer and a network.

In FIG. **5**, client computer **200** communicates with server computer **204** via network **206**. Both client computer **200** and server computer **204** use a network protocol layer to communicate with network **206**. In a preferred embodiment, network **206** is the Internet and the network protocol layers are TCP/IP. Other networks and network protocols may be used. For ease of illustration, additional hardware and software layers are not shown in FIG. **5**.

Client computer **200** includes processes, such as browser client **208** and application client **210**. In a preferred

embodiment, application client **210** is resident within client computer **200** prior to browser client **208**'s parsing of a hypermedia document as discussed below. In a preferred embodiment application client **210** resides on the hard disk or RAM of client computer **200** and is loaded (if necessary) and executed when browser client **208** detects a link to application client **210**. The preferred embodiment uses the XEvent interprocess communication protocol to exchange information between browser client **208** and application client **210** as described in more detail, below. Another possibility is to install application client **210** as a "terminate and stay resident" (TSR) program in an operating system environment, such as X-Window. Thereby making access to application client **210** much faster.

Browser client **208** is a process that a user of client computer **200** invokes in order to access various data objects, such as hypermedia documents, on network **206**. Hypermedia document **212** shown within client computer **200** is an example of a hypermedia document, or object, that a user has requested access to. In this example, hypermedia document **212** has been retrieved from a server connected to network **206** and has been loaded into, e.g., client computer **200**'s RAM or other storage device.

Once hypermedia document **212** has been loaded into client computer **200**, browser client **208** parses hypermedia document **212**. In parsing hypermedia document **212**, browser client **208** detects links to data objects as discussed above in the Background of the Invention section. In FIG. **5**, hypermedia document **212** includes an embedded program link at **214**. Embedded program link **214** identifies application client **212** as an application to invoke. In this present example, the application, namely, application client **210**, resides on the same computer as the browser client **208** that the user is executing to view the hypermedia document. Embedded program link **214** may include additional information, such as parameters, that tell application client **210** how to proceed. For example, embedded program link **214** may include a specification as to a data object that application client **210** is to retrieve and process.

When browser client **208** encounters embedded program link **214**, it invokes application client **210** (optionally, with parameters or other information) and application client **210** executes instructions to perform processing in accordance with the present invention.

An example of the type of processing that application client **210** may perform is multidimensional image visualization. Note that application client **210** is in communication with network **206** via the network protocol layer of client computer **200**. This means that application client **210** can make requests over network **206** for data objects, such as multidimensional image objects. For example, application client **210** may request an object, such as object **1** at **216**, located in server computer **204**. Application client **210** may make the request by any suitable means. Assuming network **206** is the Internet, such a request would typically be made by using HTTP in response to a HTML-style link definition for embedded program link **214**.

Assuming application client **210** has made a request for the data object at **216**, server process **218** ultimately receives the request. Server process **218** then retrieves data object **216** and transfers it over network **206** back to application client **210**. To continue with the example of a multidimensional visualization application, data object **216** may be a three dimensional view of medical data for, e.g., an embryo.

After application client **210** receives the multidimensional data object **216**, application client **210** executes instructions to display the multidimensional embryo data on the display screen to a user of the client computer **200**. The user is then able to interactively operate controls to recompute different views for the image data. In a preferred embodiment, a control window is displayed within, or adjacent to, a window generated by browser client **208** that contains a display of hypermedia document **212**. An example of such display is discussed below in connection with FIG. **9**. Thus, the user is able to interactively manipulate a multidimensional image object by means of the present invention. In order to make application client **210** integral with displays created by browser client **208**, both the browser client and the application client must be in communication with each other, as shown by the arrow connecting the two within client computer **200**. The manner of communication is through an application program interface (API), discussed below.

Browser client **208** is a process, such as NCSA Mosaic, Cello, etc. Application client **210** is embodied in software presently under development called "VIS" and "Panel" created by the Center for Knowledge Management at the University of California, San Francisco, as part of the Doyle Group's distributed hypermedia object embedding approach described in "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web," by C. Ang, D. Martin, M. Doyle; to be published in the Proceedings of Visualization 1994, IEEE Press, Washington, D.C., October 1994.

Versions and descriptions of software embodying the present invention are generally available as hyperlinked data objects from the Visible Embryo Project's World Wide Web document at the URL address "HTTP://visembryo.ucsf.edu/".

Another embodiment of the present invention uses an application server process executing on server computer **204** to assist in processing that may need to be performed by an external program. For example, in FIG. **5**, application server **220** resides on server computer **204**. Application server **220** works in communication with application client **210** residing on client computer **200**. In a preferred embodiment, application server **220** is called VRServer, also a part of Doyle Group's approach. Since server computer **204** is typically a larger computer having more data processing capabilities and larger storage capacity, application server **220** can operate more efficiently, and much faster, than application client **210** in executing complicated and numerous instructions.

In the present example where a multidimensional image object representing medical data for an embryo is being viewed, application server **220** could perform much of the viewing transformation and volume rendering calculations to allow a user to interactively view the embryo data at their client computer display screen. In a preferred embodiment, application client **210** receives signals from a user input device at the user's client computer **200**. An example of such input would be to rotate the embryo image from a current position to a new position from the user's point of view. This information is received by application client **210** and processed to generate a command sent over network **206** to application server **220**. Once application server **220** receives the information in the form of, e.g., a coordinate transformation for a new viewing position, application server **220** performs the mathematical calculations to compute a new view for the embryo image. Once the new view has been computed, the image data for the new view is sent over network **206** to application client **210** so that application client **210** can update the viewing window currently displaying the embryo image. In a preferred embodiment,

application server **220** computes a frame buffer of raster display data, e.g., pixel values, and transfers this frame buffer to application client **210**. Techniques, such as data compression and delta encoding, can be used to compress the data before transmitting over network **206** to reduce the bandwidth requirement.

It will be readily seen that application server **220** can advantageously use server computer **204**'s computing resources to perform the viewing transformation much more quickly than could application client **210** executing on client computer **200**. Further, by only transmitting the updated frame buffer containing a new view for the embryo image, the amount of data sent over network **206** is reduced. By using appropriate compression techniques, such as, e.g., MPEG (Motion Picture Experts Group) or JPEG (Joint Photographic Experts Group), efficient use of network **206** is preserved.

FIG. **6** shows yet another embodiment of the present invention. FIG. **6** is similar to FIG. **5**, except that additional computers **222** and **224** are illustrated. Each additional computer includes a process labeled "Application (Distributed)." The distributed application performs a portion of the task that an application, such as application server **220** or application client **210**, perform. In the present. example, tasks such as volume rendering may be broken up and easily performed among two or more computers. These computers can be remote from each other on network **206**. Thus, several computers, such as server computer **204** and additional computers **222** and **224** can all work together to perform the task of computing a new viewpoint and frame buffer for the embryo for the new orientation of the embryo image in the present example. The coordination of the distributed processing can be performed at client computer **200** by application client **210**, at server computer **204** by application server **220**, or by any of the distributed applications executing on additional computers, such as **222** and **224**. In a preferred embodiment, distributed processing is coordinated by a program called "VIS" represented by application client **210** in FIG. **6**.

Other applications of the invention are possible. For example, the user can operate a spreadsheet program that is being executed by one or more other computer systems connected via the network to the user's client computer. Once the spreadsheet program has calculated results, those results may be sent over the network to the user's client computer for display within the hypermedia document on the user's client computer. In this way, computer systems located remotely on the network can be used to provide the computing power that may be required for certain tasks and to reduce the data bandwidth required by only transmitting results of the computations.

Another type of possible application of this invention would involve embedding a program which runs only on the client machine, but which provides the user with more functionality than exists in the hypermedia browser alone. An example of this is an embedded client application which is capable of viewing and interacting with images which have been processed with Dr. Doyle's MetaMAP invention (U.S. Pat. No. 4,847,604). This MetaMAP process uses object-oriented color map processing to allow individual color index ranges within paletted images to have object identities, and is useful for the creation of, for example, interactive picture atlases. It is a more efficient means for defining irregular "hotspots" on images than the ISMAP function of the World Wide Web, which uses polygonal outlines to define objects in images. A MetaMAP-capable client-based image browser application can be embedded, together with an associated image, within a hypermedia document, allowing objects within the MetaMAP-processed image to have URL addresses associated with them. When a user clicks with a mouse upon an object within the MetaMAP-processed image, the MetaMAP client application relays the relevant URL back to the hypermedia browser application, which then retrieves the HTML file or hypermedia object which corresponds to that URL.

The various processes in the system of the present invention communicate through a custom API called Mosaic/External Application Program Interface MEAPI. The MEAPI set of predefined messages includes those shown in Table I.

TABLE I

| Message Function | Message Name |
|---|---|
| Messages from server to client: | |
| 1. Server Update Done | XtNrefreshNotify |
| 2. Server Ready | XtNpanelStartNotify |
| 3. Server Exiting | XtNpanelExitNotify |
| Messages from client to server: | |
| 4 Area Shown | XtNmapNotify |
| 5 Area Hidden | XtNunmapNotify |
| 6 Area Destroyed | XtNexitNotify |

The messages in Table I are defined in the file protocol_lib.h in Appendix B. The functions of the MEAPI are provided in protocol_lib.c of Appendix B. Thus, by using MEAPI a server process communicates to a client application program to let the client application know when the server has finished updating information, such as an image frame buffer, or pixmap (Message **1**); when the server is ready to start processing messages (Message **2**) and when the server is exiting or stopping computation related to the server application program.

For client to server communications, MEAPI provides for the client informing the server when the image display window area is visible, when the area is hidden and when the area is destroyed. Such information allows the server to decide whether to allocate computing resources for, e.g., rendering and viewing transformation tasks where the server is running an application program to generate new views of a multi dimensional object. Source code for MEAPI fundamental functions such as handle_client_msg, register_client, register_client_msg_callback and send_client_msg may be found in protocol_lib.c as part of the source code in Appendix B.

Next, a discussion of the software processes that perform parsing of a hypermedia document and launching of an application program is provided in connection with Table II and FIGS. **7A**, **7B**, **8A** and **8B**.

Table II, below, shows an example of an HTML tag format used by the present invention to embed a link to an application program within a hypermedia document.

TABLE II

```
<EMBED
    TYPE = "type"
    HREF = "href"
    WIDTH = width
    HEIGHT = height
>
```

As shown in Table II, the EMBED tag includes TYPE, HREF, WIDTH and HEIGHT elements. The TYPE element

is a Multipurpose Internet Mail Extensions (MIME) type. Examples of values for the TYPE element are "application/x-vis" or "video/mpeg". The type "application /x-vis" indicates that an application named "x-vis" is to be used to handle the object at the URL specified by the HREF. Other types are possible such as "application/x-inventor", "application/postscript" etc. In the case where TYPE is "application/x-vis" this means that the object at the URL address is a three dimensional image object since the program "x-vis" is a data visualization tool designed to operate on three dimensional image objects. However, any manner of application program may be specified by the TYPE element so that other types of applications, such as a spreadsheet program, database program, word processor, etc. may be used with the present invention. Accordingly, the object reference by the HREF element would be, respectively, ;a spreadsheet object, database object, word processor document object, etc.

On the other hand, TYPE values such as "video/mpeg", "image/gif", "video/x-sgi-movie", etc. describe the type of data that HREF specifies. This is useful where an external application program, such as a video player, needs to know what format the data is in, or where the browser client needs to determine which application to launch based on the data format. Thus, the TYPE value can specify either an application program or a data type. Other TYPE values are possible. HREF specifies a URL address as discussed above for a data object. Where TYPE is "application/x-vis" the URL address specifies a multi-dimensional image object. Where TYPE is "video/mpeg" the URL address specifies a video object.

WIDTH and HEIGHT elements specify the width and height dimensions, respectively, of a Distributed Hypermedia Object Embedding (DHOE) window to display an external application object such as the three dimensional image object or video object discussed above

FIG. 7A is a flowchart describing some of the functionality within the HTMLparse c file of routines. The routines in HTMLparse.c perform the task of parsing a hypermedia document and detecting the EMBED tag. In a preferred embodiment, the enhancements to include the EMBED tag are made to an HTML library included in public domain NCSA Mosaic version 2.4. Note that much of the source code in is pre-existing NCSA Mosaic code. Only those portions of the source code that relate to the new functionality discussed in this specification should be considered as part of the invention. The new functionality is identifiable as being set off from the main body of source code by conditional compilation macros such as "#ifdef . . . #endif" as will be readily apparent to one of skill in the art.

In general, the flowcharts in this specification illustrate one or more software routines executing in a computer system such as computer system **1** of FIG. 1. The routines may be implemented by any means as is known in the art. For example, any number of computer programming languages, such as "C", Pascal, FORTRAN, assembly language, etc., may be used. Further, various programming approaches such as procedural, object oriented or artificial intelligence techniques may be employed.

The steps of the flowcharts may be implemented by one or more software routines, processes, subroutines, modules, etc. It will be apparent that each flowchart is illustrative of merely the broad logical flow of the method of the present invention and that steps may be added to, or taken away from, the flowcharts without departing from the scope of the invention. Further, the order of execution of steps in the flowcharts may be changed without departing from the scope of the invention. Additional considerations in implementing the method described by the flowchart in software may dictate changes in the selection and order of steps. Some considerations are event handling by interrupt driven, polled, or other schemes. A multiprocessing or multitasking environment could allow steps to be executed "concurrently." For ease of discussion the implementation of each flowchart may be referred to as if implemented in a single "routine".

The modifications to NCSA Mosaic version 2.4 software files HTMLparse.c, HTMLformat.c, HTMLwidget.c and HTML.c will next be discussed, in turn.

Returning to FIG. **7**, it is assumed that a hypermedia document has been obtained at a user's client computer and that a browser program executing on the client computer displays the document and calls a first routine in the HTMLparse.c file called "HTMLparse". This first routine, HTMLparse, is entered at step **252** where a pointer to the start of the document portion is passed. Steps **254**, **256** and **258** represent a loop where the document is parsed or scanned for HTML tags or other symbols. While the file HTMLparse.c includes routines to handle all possible tags and symbols that may be encountered, FIG. **7A**, for simplicity, only illustrates the handling of EMBED tags.

Assuming there is more text to parse, execution proceeds to step **256** where routines in HTMLparse.c obtain the next item (e.g., word, tag or symbol) from the document. At step **258** a check is made as to whether the current tag is the EMBED tag. If not, execution returns to step **254** where the next tag in the document is obtained. If, at step **258**, it is determined that the tag is the EMBED tag, execution proceeds to step **260** where an enumerated type is assigned for the tag. Each occurrence of a valid EMBED tag specifies an embedded object. HTMLParse calls a routine "get_mark" in HTMLparse.c to put sections of HTML document text into a "markup" text data structure. Routine get_mark, in turn, calls ParseMarkType to assign an enumerated type. The enumerated type is an identifier with a unique integer associated with it that is used in later processing described below.

Once all of the hypermedia text in the text portion to be displayed has been parsed, execution of HTMLparse.c routines terminates at step **262**.

FIG. **7B** is a flowchart of routines in file HTMLformat.c to process the enumerated type created for the EMBED tag by routines in HTMLparse.c. In the X-Window implementation of a preferred embodiment, the enumerated type is processed as if it is a regular Motif/XT widget. For details on X-Window development see, e.g., "Xlib Programming Manual," "X Toolkit Intrinsics Programming Manual" and "Motif Programming Manual" published by O'Reilly & Associates, Inc. HTMLformat is entered at step **270** where a pointer to the enumerated type to process is passed.

At step **272** the parameters of the structure are initialized in preparation for inserting a DrawingArea widget on an HTML page. This includes providing values for the width and height of a window on the display to contain an image, position of the window, style, URL of the image object, etc. Various codes are also added by routines in HTMLformat.c (such as TriggerMarkChanges) to insert an internal representation of the HTML statement into an object list maintained internally by the browser. In the X-Window application corresponding to the source code of Appendix A, the browser is NCSA Mosaic version 2.4.

FIG. **8A** is a flowchart for routine HTMLwidget. HTMLwidget creates display data structures and launches an external application program to handle the data object specified by the URL in the EMBED tag.

HTMLwidget is entered at step **280** after HTMLformat has created the internal object representation of the EMBED tag. HTMLwidget is passed the internal object and performs its processing on the object. At step **282** the DrawingArea widget is created according to the type of the internal representation, from HTMLformat, specified in the internal object. Similarly, at step **284** a pixmap area for backing storage is defined.

At step **286** a check is made as to whether the type attribute of the object, i.e., the value for the TYPE element of the EMBED tag, is an application. If so, step **290** is executed to launch a predetermined application. In a preferred embodiment an application is launched according to a user-defined list of application type/application pairs. The list is defined as a user-configurable XResource as described in "Xlib Programming Manual." An alternative embodiment could use the MIME database as the source of the list of application type/application pairs. The routine "vis_start_external_application" in file HTMLformat.c is invoked to match the application type and to identify the application to launch.

The external application is started as a child process of the current running process (Mosaic), and informed about the window ID of the DrawingArea created in HTMLformat. The external application is also passed information about the ID of the pixmap, the data URL and dimensions. Codes for communication such as popping-up/iconifying, start notification, quit notification and refresh notification with external applications and DrawingArea refreshing are also added. Examples of such codes are (1) "setup/start" in vis_register_client and vis_get panel_window in HTMLwidgets.c; (2) "handle messages from external applications" in vis_handle panel_msg in HTMLwidgets.c; (3) "send messages to external applications" in vis_send_msg in HTMLwidgets.c; (4) "terminate external applications" in vis_exit in HTMLwidgets.c which calls vis_send_msg to send a quit message; and (5) "respond to refresh msgs" in vis_redraw in HTMLwidgets.c.

If, at step **286**, the type is determined not to be an application object (e.g., a three dimensional image object in the case of application "x-vis") a check is made at step **288** to determine if the type is a video object. If so, step **292** is executed to launch a video player application. Parameters are passed to the video player application to allow the player to display the video object within the DrawingArea within the display of the portion of hypermedia document on the client's computer. Note that many other application objects types are possible as described above.

FIG. **8B** is a flowchart for routine HTML. Routine HTML takes care of "shutting down" the objects, data areas, etc. that were set up to launch the external application and display the data object. HTML is entered at step **300** and is called when the display or other processing of the EMBED tag has been completed. At step **302** the display window is removed and the memory areas for the pixmap and internal object structure is made free for other uses. Completion of processing can be by user command or by computer control.

The present invention allows a user to have interactive control over application objects such as three dimensional image objects and video objects. In a preferred embodiment, controls are provided on the external applications' user interface. In the case of a VIS/panel application, a process, "panel" creates a graphical user interface (GUI) thru which the user interacts with the data. The application program, VIS, can be executing locally with the user's computer or remotely on a server, or on one or more different computers, on the network. The application program updates pixmap data and transfers the pixmap data (frame image data) to a buffer to which the browser has access. The browser only needs to respond to the refresh request to copy the contents from the updated pixmap to the DrawingArea. The Panel process sends messages as "Msg" sending performed by routines such as vis_send_msg and vis_handle panel_msg to send events (mousemove, keypress, etc.) to the external application.

FIG. **9** is a screen display of the invention showing an interactive application object (in this case a three dimensional image object) in a window within a browser window. In FIG. **9**, the browser is NCSA Mosaic version 2.4. The processes VIS, Panel and VRServer work as discussed above. FIG. **9** shows screen display **356** Mosaic window **350** containing image window **352** and a portion of a panel window **354**. Note that image window **352** is within Mosaic window **350** while panel window **354** is external to Mosaic window **350**. Another possibility is to have panel window **354** within Mosaic window **350**. By using the controls in panel window **354** the user is able to manipulate the image within image window **352** in real time do perform such operations as scaling, rotation, translation, color map selection, etc. In FIG. **9**, two Mosaic windows are being used to show two different views of an embryo image. One of the views is rotated by six degrees from the other view so that a stereoscopic effect can be achieved when viewing the images. Communication between Panel and VIS is via "Tooltalk" described in, e.g., "Tooltalk 1.1.1 Reference Manual," from SunSoft.

FIG. **10** is an illustration of the processes VIS, Panel and VRServer discussed above. As shown in FIG. **10**, the browser process, Mosaic, communicates with the Panel process via inter-client communication mechanisms such as provided in the X-Window environment. The Panel process communicates with the VIS process through a communications protocol (ToolTalk, in the preferred embodiment) to exchange visualization command messages and image data. The image data is computed by one or more copies of a process called VRServer that may be executing on remote computers on the network. VRServer processes respond to requests such as rendering requests to generate image segments. The image segments are sent to VIS and combined into a pixmap, or frame image, by VIS. The frame image is then transferred to the Mosaic screen via communications between VIS, Panel and Mosaic. A further description of the data transfer may be found in the paper "Integrated Control of Distributed Volume Visualization Through the World-Wide-Web," referenced above.

In the foregoing specification, the invention has been described with reference to a specific exemplary embodiment thereof. It will, however, be evident that various modifications and changes may be made thereunto without departing from the broader spirit and scope of the invention as set forth in the appended claims. For example, various programming languages and techniques can be used to implement the disclosed invention. Also, the specific logic presented to accomplish tasks within the present invention may be modified without departing from the scope of the invention. Many such changes or modifications will be readily apparent to one of ordinary skill in the art. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense, the invention being limited only by the provided claims.

What is claimed is:

**1**. A method for running an application program in a computer network environment, comprising:

providing at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment;

executing, at said client workstation, a browser application, that parses a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and for responding to predetermined text formats to initiate processing specified by said text formats; utilizing said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document, wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document, and wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable interactive processing of said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.

2. The method of claim 1, wherein said executable application is a controllable application and further comprising the step of:

interactively controlling said controllable application on said client workstation via inter-process communications between said browser and said controllable application.

3. The method of claim 2, wherein the communications to interactively control said controllable application continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched.

4. The method of claim 3, wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes the following substeps:

issuing, from the client workstation, one or more commands to the network server,

executing, on the network server, one or more instructions in response to said commands;

sending information from said network server to said client workstation in response to said executed instructions; and processing said information at the client workstation to interactively control said controllable application.

5. The method of claim 4, wherein said additional instructions for controlling said controllable application reside on said client workstation.

6. A computer program product for use in a system having at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment, the computer program product comprising:

a computer usable medium having computer readable program code physically embodied therein, said computer program product further comprising:

computer readable program code for causing said client workstation to execute a browser application to parse a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and to respond to predetermined text formats to initiate processes specified by said text formats;

computer readable program code for causing said client workstation to utilize said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document, wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document, and wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable interactive processing of said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.

7. The computer program product of claim 6, wherein said executable application is a controllable application and further comprising:

computer readable program code for causing said client workstation to interactively control said controllable application on said client workstation via inter-process communications between said browser and said controllable application.

8. The computer program product of claim 7, wherein the communications to interactively control said controllable application continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched.

9. The computer program product of claim 8, wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes:

computer readable program code for causing said client workstation to issue, from the client workstation, one or more commands to the network server;

computer readable program code for causing said network server to execute one or more instructions in response to said commands;

computer readable program code for causing said network sever to send information to said client workstation in response to said executed instructions; and

computer readable program code for causing said client workstation to process said information at the client workstation to interactively control said controllable application.

10. The computer program product of claim 9, wherein said additional instructions for controlling said controllable application reside on said client workstation.

* * * * *

E

Approved For Release 2004/12/20 : CIA-RDP80B01554R000100040001-0

# Attachment E

# Press Coverage

4-02-20 - Attached

CNET tech sites: Price comparisons | Product reviews | Tech news | Dow

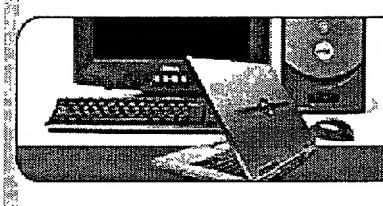
cnet NEWS.COM TECH NEWS FIRST

FRONT PAGE | ENTERPRISE SOFTWARE | ENTERPRISE HARDWARE | SECURITY | NETWORKING | PERSONAL TECH

SAVED STORIES

SEAR

advertisement



Technology for Every Need. Deals for Every Budget.
Start saving now with new specials every week!

Mor

Dell | S

NEWS.COM SPECI

# PATENT POLITICS

STAKING A CLAIM

## Rivalries set aside in defense of Internet Explorer

By Paul Festa
Staff Writer, CNET News.com
September 25, 2003, 4:00AM PT

**During a recent meeting held at Macromedia's San Francisco headquarters, Silicon Valley companies asked a familiar question: What to do about Microsoft?**

But the strategy event, sponsored by the World Wide Web Consortium, differed significantly from so many others, at which participants have typically gathered to oppose the software giant's power. This time, Microsoft was the guest of honor.

### News.context

**What's new:**
A verdict against Microsoft in a patent infringement case is threatening to force significant changes to the Web's fundamental language, HTML.

**Bottom line:**
Microsoft's competitors fear they may be targeted next and that an enjoined IE browser would be prohibited from running their software plug-ins without awkward

HIGH IMPACT

"There's no doubt that there are some people who are happy to see Microsoft get nailed for anything," said Dale Dougherty, a vice president at computer media company O'Reilly & Associates. "But for those of us who are part of the Web, we wanted the browser to be on every desktop. And if it has to be a Microsoft browser, OK."

What a difference a patent suit makes. With one staggering loss at the hands of a federal court jury

Eolas has attracted the patent spotlight the historic proportions of its $521 millic judgment against Microsoft, the power a of that target, and how its claims could rest of the Web. But other threats loom array of technologies many Web develo for granted. Here's a sample of Web-rel patents worth watching:

### Yahoo: Web search

Yahoo, through its proposed acquisition Overture, looks likely to gain a formidat search patent arsenal. Before news of th broke, Overture had already fired shots of lawsuits against Google and FindWha

"We'll add...Overture's impressive intell property portfolio of both algorithmic an sponsored search patents. These are so key reasons we have opted to acquire O --*Terry Semel, CEO, Yahoo (July 14, 20* Overture to a patent war?

### Acacia Research and Friskit: streaming media

Acacia Research has successfully presse claims against an array of Web sites and networks. While many of these are in th pornography business, the company say negotiations with the biggest mainstrea

technology alternatives. As a result, Microsoft is finding some unlikely friends who have come to depend on its browser for distribution of their products.

**For more info:**
More news on patents

in Chicago, Microsoft has won the support--if not the sympathy--of nearly the entire software industry, from standards organizations to corporate rivals that are rushing to defend the company's Internet Explorer browser.

To some competitors and partners who have long been chafed by Microsoft's dominance, the verdict in the patent infringement lawsuit by one-man software company Eolas may initially have seemed an overdue victory--and one that achieved what the U.S. Department of Justice and the courts had failed to accomplish in regulating Microsoft under federal antitrust laws.

Instead, the verdict is increasingly interpreted as a potentially crushing burden on the Web, threatening to force significant changes to its fundamental language, HTML. Microsoft's competitors fear that Eolas' lawyers will target them next, and its partners--such as Macromedia and Sun Microsystems--worry that an enjoined IE browser would be prohibited from running their software plug-ins without awkward technology alternatives.

The result has been a complex shift of industry dynamics that has turned many traditional alliances and rivalries upside down, prompting long-suffering competitors in the browser market to side with archrival Microsoft. At the same time, as the Eolas case has progressed, critics have portrayed company founder and sole employee Mike Doyle as an opportunist, despite his claims to be acting on behalf of the Web against a rapacious captor.

**Related news**
Microsoft--a security risk? ▸
The company's dominance poses a threat to national security, says a report.

"If he has some altruistic motives here, I think it's time to step forward and describe them in more detail," one technologist for an IE competitor said. "Instead, his lawyers have said that everyone in the browser business should talk to them about getting a license. Maybe something good can come of this, but all in all, it's a very frustrating way to be saved."

Doyle counters that he wants only to correct Microsoft's misdeeds and liberate the masses from the oppression of Bill Gates' vast empire. "We, because of this legal victory, will be able to reinstate a whole new area for competitive development in the software industry. The developers and the competing entities out there to Microsoft, I think, will come to find that we are more of a friend than a threat," he said.

Microsoft might still pull out a victory at the appellate level. Moreover, even if Eolas' patent is upheld, the rest of the software industry may very well go with Microsoft's workarounds rather than face the prospect of abandoning development for the universally distributed IE.

Filed in 1999, the Eolas case

providers as well. Most worrisome to the industry is the broad scope of the comp streaming media patents.

In addition, Friskit in July brought suit a RealNetworks and Listen.com for violati streaming media patents; Real closed it acquisition of Listen the following mont

"All the methods we have looked at for audio and video over the Internet are c our patents."
*--Rob Berman, senior vice president and counsel, Acacia (Sept. 23, 2003)*
Patent holder unplugs porn network
Broad patents on streaming media uphe
Patent suit hits Real, Listen.com
Streaming patent claims go to court

### InterTrust Technologies: digital rights management

The DRM company claims that Microsoft infringed on its copy-protection patents ranging from Xbox to Windows. The tec crucial to the dissemination of copyright over the Web.

"InterTrust's core set of inventions--star early to mid-1990s--anticipated what ot Microsoft, now realize in 2002 is very im InterTrust is prepared and has the reso this all the way through."
*--Ed Fish, executive vice president, Inte 7, 2002)*
Microsoft loses key patent ruling
InterTrust broadens suit against Microsc
Microsoft patent dispute heats up

### Amazon: Various technologies

Amazon has amassed a considerable pa portfolio and has kept busy expanding a enforcing it. Its claims cover ordering fo chat; second-hand sales; one-click orde auctioning Web advertisements. The cor patent ambitions are so broad that som are scratching their heads over its inten

"Now, when people come up with new ic they're more apt to try to create a pater for things to evolve. It may be somethir want to do or (that they want to) sell la
*--David Halprin, Internet analyst and co (March 24, 2003)*
Amazon wins patent for ordering forms
Patent company sues Amazon
Amazon wins retail chat patent
Amazon patent bid targets used goods
Amazon makes bid for Web-ad patent
Amazon, Barnes & Noble settle patent s

READER RESOURCES

drew international attention last month, when a U.S. District Court ruled that Microsoft's IE browser violated Eolas' Patent No. 5,838,906. The patent, filed on Oct. 17, 1994, and granted Nov. 17, 1998, covers a system that launches an application within a Web page.

The jury found that IE infringed on the patent through its inclusion of Microsoft's ActiveX technology, which Web authors use to launch and run plug-in applications such as Java applets, Adobe Acrobat documents and Macromedia Flash movies. Without ActiveX, the Microsoft browser would be unable to fully render a significant proportion of pages on the Web as well as on many corporate intranets.

This is why Doyle says his suit, if successful, will right many wrongs Microsoft inflicted in crushing upstart browser rival Netscape Communications. When Netscape launched in 1994, its founders saw the Web browser as a potential end run around the Windows juggernaut. Rather than having to code applications to work with Windows, the computing world could write to the open standard of the Web, Netscape and its investors reasoned--and the operating system would therefore be reduced to a mere commodity from a multibillion-dollar, Microsoft-controlled toll booth.

But Microsoft's yearslong assault on the browser market reduced Netscape from a standard-bearer (with more than 80 percent of the market) to a neglected unit of AOL Time Warner, which recently spun off its browser division as a nonprofit foundation. Industry veterans say there is every reason to believe that Microsoft will survive this challenge as well.

Given the daunting odds in any challenge to Microsoft, Doyle believes that his struggle exceeds biblical proportions. He said the often-cited comparisons to David and Goliath don't go far enough in conveying the ambition and travails of his quest, which he believes could reverse Microsoft's victory in the so-called browser war and break its control over much of the digital world.

**News.commentary**
Guess what? Microsoft won
The software giant is stronger than ever, says Charles Cooper.

"'David vs. Goliath' minimizes what we're doing," Doyle said. "Microsoft's use of our technology has been to put others out of business and to restrict much of the potential of the Web. And that is limiting our ability to get the full value for things that we created."

Many software makers might have supported Eolas when Microsoft was more vulnerable to browser competition years ago, but they now say Doyle's efforts have come too late. Since IE rose to market dominance, many software companies have come to rely on the browser's plug-ins for their businesses.

"We're no big fan of Microsoft, but I'm a big fan of the Web," said Dougherty, who is in charge of online publishing at O'Reilly and testified on behalf of Microsoft in its recent patent trial. "What worries people is that this is the first successful patent offense on the Web,

Eolas' patent
U.S. Patent and Trademark Office

W3C's Patent Advisory Group
W3C

Public discussion group on the Eolas pat
W3C

Q&A about University of California's role
Eolas patent suit
UC Office of the President

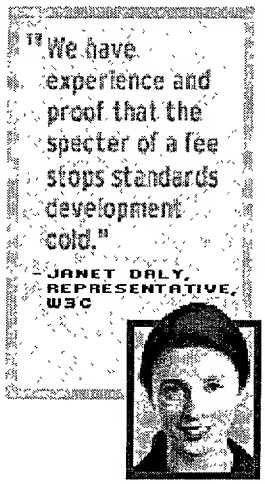
Ray Ozzie's notes on "prior art," or a pre
technology that would invalidate a pater
Ray Ozzie's Weblog

**RELATED NEWS**

Eolas says it would settle over IE
Eolas suit may spark HTML changes
IE patent endgame detailed
Will Microsoft tweak IE?
Will browser verdict snare others?
Microsoft ordered to pay $521 million

**Editors:** Mike Yamamoto, Karen Said
**Copy editor:** Zoë Barton
**Design:** Pam Dore
**Production:** Meghan McDowell

and lots of other things could be coming."




Some browser application makers look likely to come up with relatively easy workaround solutions. Adobe, for example, already offers two methods for reading a PDF (Portable Document Format) file that is posted on a Web page: One requires the IE technology to open a document within the Web browser, but the other shows the document in its Acrobat reader and therefore probably steers clear of the patent issue.

For others, the end of IE plug-ins could be disastrous. "Macromedia is clearly the most vulnerable," said Richard Smith, a noted computer security analyst and a participant in a W3C online discussion about the Eolas patent.

"If you look at embedded content in Web pages--that is, plug-ins--Flash has to be No. 1 by a mile," he said. "In my reading of the patent, the ways that Macromedia operates--and the things that it does--make it seem that it would fall under the patent."

Macromedia notes that it is hardly the only company with technologies that rely on plug-ins, pointing to applets that are written in Sun's Java programming language as just one example of other software that would be comparably affected by the patent issue.

As far as Doyle is concerned, none of the recently proposed workarounds would insulate companies from Eolas' patent claims. That argument has led Microsoft to return an accusation often made against itself, charging that Doyle is spreading "FUD"--fear, uncertainty and doubt--while he shores up support for his case.

"That's sort of a transparent and self-serving attempt by Mr. Doyle to put a cloud of uncertainty over the industry with respect to the breadth and scope of the patent," Microsoft spokesman Jim Desler said. "Any reasonable reading of the patent, as well as what Eolas said itself about the patent during trial, shows that the modest changes we're considering would avoid any infringement."

Whatever the scope of his patent, which is co-owned by the University of California, Doyle could theoretically wind up with the power to grant Macromedia, Adobe, Sun and the rest of the plug-in makers a simple alternative, which could take the form of an Eolas browser or an affordable license.

The prospect of having such a basic necessity as running plug-ins subject to the whim of Eolas has the industry in a near panic--not least among those organizations whose rules restrict or ban the use of patented technologies, such as open-source browser makers and the W3C.

Groups that advocate software that has open-source code say their licenses prohibit them from including patented technologies. The W3C in March reaffirmed its opposition to the use of royalty-encumbered technologies, after a lengthy public battle that ended in a near-ban.

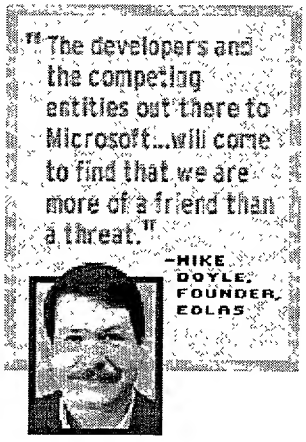
"We have experience and proof that the specter of a fee stops standards development cold," W3C representative Janet Daly said. "It doesn't even have to be a firm guarantee. All you need is a little bit of fear, uncertainty and doubt that a developer is going to be slapped with a licensing fee, and the developer will leave that technology alone."

**Endgame**

Ultimately, the fight between Microsoft and Eolas represents a larger battle between those who believe in patents and those who assert that these have no place in software. (Microsoft's position in this particular skirmish presents a certain irony, considering its own formidable patent portfolio.)

Doyle, who recently signaled his willingness to settle the case with Microsoft, maintains that antipatent licenses, policies and attitudes are paving the road to obsolescence.

"The standards people have to recognize that the creators of technology will continue to be able to acquire patents to protect their intellectual property, regardless of whatever standards they try to force down people's throats," he said. "They're just in denial, institutionally."




Doyle's fervent belief in the patent system may be in his genes. His grandfather used patents to protect more than 60 inventions in the papermaking industry, including a 1918 "Paper-Machine," a 1949 "Pulp Drainer" and a 1965 "Apparatus for Draining Fibrous Material."

"I'm just an inventor, and Eolas is a company whose mission is invention," Doyle said. "And we want to be able to make a living at invention, and build a business around invention. Why am I so focused on that? My grandfather was an inventor, and he made a living inventing printing processes that are still used today." ■

Send us news tips | Contact us | Corrections | Privacy policy | XML | Contact licensing | Get News.com mobile | New

FRONT PAGE | ENTERPRISE SOFTWARE | ENTERPRISE HARDWARE | SECURITY | NETWORKING | PERSONAL TECH

**Featured services:** Tech Jobs | Free Magazine Trial | Learning CDs | Hot Downloads | Clearance Center

CNET Networks:

NEWS

# Microsoft's patent loss rattles tech community

Company says it is responding by making changes to Internet Explorer

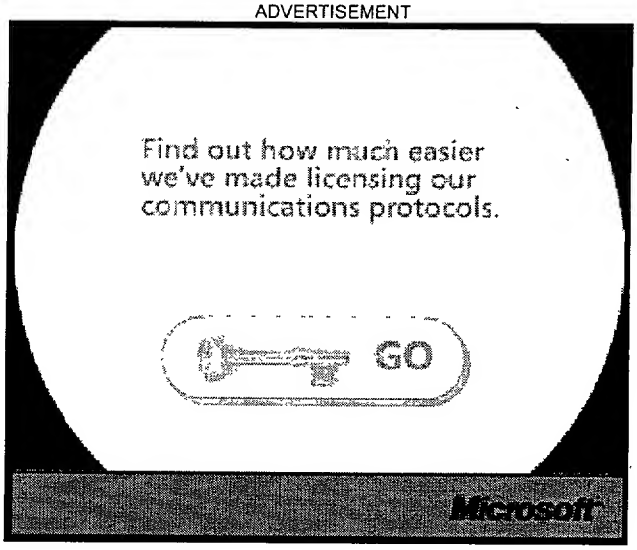
**By Paul Roberts, IDG News Service** September 03, 2003

NEWS
Top Stories
* Software l
must evol

Companies with products that work on the Internet are slowly waking up to the broad implications of a recent judgement against software behemoth Microsoft Corp. in a patent infringement case.

The $520 million award to Eolas Technologies Inc. of Chicago and the University of California (UC) stemmed from a 1999 lawsuit in which Eolas and UC charged Microsoft with infringing on a 1998 patent owned by the university and licensed to Eolas. However, the verdict could spell trouble for a wide range of popular Web-based products and services, experts agree.

ADVERTISEMENT




That patent, U.S. number 5,838,906, was developed by Eolas president Michael Doyle at the University of California in San Francisco and covers technology that enables small computer programs, often referred to as "applets" or "plug-ins," to be embedded in Web pages and interacted with through Web browsers like Microsoft's Internet Explorer.

In response to the judgement against it, Microsoft said last week that it will be making changes to Internet Explorer (IE) that may affect a "large number of existing Web pages," according to a statement by the World Wide Web Consortium (W3C).

In addition to pursuing post-trial motions against Eolas, Microsoft is also evaluating what changes may be necessary and will not comment on its work, according to company spokesman Jim Desler. The Redmond, Wash., company stands by its claims that it did not infringe on the Eolas patent, but will work to minimize the effect on customers of changes to IE and is cooperating with the W3C to coordinate that effort, he said.

Computer security experts initially hailed the announcement, speculating that the ruling might spell the end of Microsoft's ActiveX controls, notoriously insecure software components that allow software developers to integrate specialized functionality with Web pages.

But technology and legal experts agree that the ruling could affect a wide range of technology companies with products that interact with Web browsers, or services that rely on customer interaction through Web browsers.

"Fundamentally, (the Eolas patent) describes a way of implementing plug-ins in a Web browser," said Richard Smith, an independent technology expert in Boston. "People who use plug-ins like (Macromedia Inc.'s) Flash or Java applets are covered by the Eolas patent," he said.

Macromedia, which distributes a free plug-in to view Macromedia Flash files, did not respond to requests for comment.

Real Software Inc., which makes multimedia software that can be played through Web browsers, said it could not immediately comment on the ruling.

However, the W3C is concerned about the implications of Eolas' patent claim, according to Janet Daly, the organization's head of communications.

"There certainly are concerns whenever patent issues ... appear to be relevant to basic technology. That gets the attention of the W3C membership," she said.

Past patent claims, such as those affecting the P3P (Platform for Privacy Preferences ) standard, have stopped development or the implementation of development standards, she said.

As in that case, the W3C has legal and technology experts analyzing the Eolas patent and the legal decisions that led to the company's court victory over Microsoft, according to Daly. That analysis could take six months or more, but the group will make its findings public once they are known, she said.

Among other things, the group is trying to determine whether any of its published standards infringe on the Eolas patent, Daly said.

In the meantime, technologists and executives who feel they may have products that infringe on Eolas' patent are following the post-trial motions closely and hoping for some word about how Eolas and the University of California will proceed.

One of those is Hector Santos, president and chief technology officer of Santronics Software Inc. of Homestead, Florida, which sells BBS (bulletin board system) software. Santos learned of the Eolas case after the $520 million verdict was announced, but became concerned about the implications for his three person company after reading more about the Eolas patent.

"As I learned more about it and understood more about what these guys patented and what it means, the more I felt like 'This claim is pretty broad!'," Santos said.

"The idea of remote client-server applets activated by a remote hosting server has been around for a while and we do it with our own technology," he said.

In particular, a "chat" feature in Santronics' Wildcat software uses a Java application (or applet) that may violate Eolas' patent, he said.

Santos is reviewing his product's code and functionality carefully in light of the suit.

"My lawyer gave the advice, 'Start reviewing what you've got,'" he said.

Santos feels that there is plenty of evidence that his company's product used Eolas' patented techniques before Doyle filed for his patent in 1994 -- an argument known as "prior art" that can be used to defuse patent infringement claims. If anything, Santos is surprised that Microsoft wasn't able to successfully use such claims in its own defense.

One of the problems may be that technologists routinely underestimate the reach of patents, according to Smith.

"Technology people don't understand what patents are and they make big claims, like, 'Oh, it's just like this or that. There's prior art.' But there was none produced by Microsoft," Smith said.

Assuming the Eolas decision stands, the fact that Microsoft apparently could produce no prior art will only serve to strengthen the patent, according to attorney Douglas Kline, chairman of the patent and intellectual property group at Boston law firm Testa, Hurwitz & Thiebeault LLP.

"Microsoft would know better than anybody what they were working on when the patent was filed," Kline said.

"To the extent they did exhibit (prior art), the jury disagreed. So if Microsoft couldn't prove that their own activity didn't render a patent invalid, it could be difficult for anyone else to prove it," he said.

Companies like Macromedia, Real and Apple Computer Inc. should all be on notice following the Eolas ruling, Kline said.

"If they didn't know about this patent before, they do now. And they have guidance about what one court thinks (the patent) means," he said.

Under patent law, Eolas and the University of California are free to go after technology companies as well as "end users" of that technology, according to attorney Jim Gatto, co-head of the intellectual property group at Mintz, Levin, Cohn, Ferris, Glovsky and Popeo PC.

Typically, however, small companies will target one or two large companies, collecting significant damages and enforcing their patent rights, he said.

The University of California is not aware of any plans to pursue parties other than Microsoft, but UC spokesman Trey Davis referred questions about legal strategies to Eolas' attorney, Martin Lueck, who did not respond to repeated requests for comment.

Regardless of what happens in the Eolas case, industry watchers should expect to see more and bigger patent cases in the future, Gatto said. The size of the judgement against Microsoft, which is one of the largest ever in terms of monetary damages, and the increasing importance of intellectual property to companies' bottom lines will drive an interest in pursuing big patent cases, he said.

In addition, the Eolas case may send a message to small companies that they can successfully defend patents in court, even when pitted against much larger companies with limitless resources like Microsoft, Gatto said.

SPONSORED WHITE PAPERS

**Cisco** - SAFE: A Security Blueprint for Enterprise Networks
**EMC** - FREE PAPER: Get "NETWORKED STORAGE BUYERS GUIDE to PAIN RELIEF" now!
**Neoteris** - See why Neoteris is the SSL VPN that analysts and security experts choose!
**Remedy** - Manage Change requests and Cut Asset-related Costs
**Remedy** - Get your FREE Remedy Asset Management Whitepaper today!

Search the IDG White Paper Library:
Search

SPONSORED LINKS

**HP - Go wireless with HP Wireless Solutions Proof of Concept.**
**Intel - There is measurable ROI for wireless LAN deployments. Tune in and learn.**
**InfoWorld Special Report - Networking Tips for Small-to-Med Businesses sponsored by Network Associates**
**CA - Unicenter(R) Infrastructure Management Software, ca.com**
**HP - Win an HP iPaq now on the Business Resource Center!**

INFOWORLD MARKETPLACE

**RFID/Smart Label Printing White Paper from Zebra** - Learn about how smart labels can help prevent asset loss, track shipments, and process customer transactions, and see how RFID technology could help your business.

**AT&T-Cisco Portal Examines IP VPN Services.** - The IP VPN Portal from AT&T and Cisco Systems features numerous resources and tools, including a Webcast on how to increase productivity, lower costs and extend the power of your network.

**Bring the Mainframe to the Web with Xbridge** - Web enable your mainframe with Xbridge Host Data Connect. You can easily integrate host data through your web applications. And, this can be done without rewriting your host applications. Bring the power of the mainframe to the web with Xbridge.

**Intuit Track-It! Help Desk Software** - Intuit IT Solutions provides Track-It! - the leading help desk software solution for call tracking, problem resolution, employee & customer self-help, remote control, asset management, LAN/PC auditing, and electronic software distribution. Free demo

**We'll help with the ebiz stuff. You play golf.** - Need time to work on your short game? Talk to UBICS, providers of IT and ebusiness services to Global 1000 companies. App. dev., Web services, Network assessment. UBICS - funny name, serious clients.

**>> BUY A LINK NOW**



Copyright © 2003, Reprints, Permissions, Licensing

CNET tech sites: | Price comparisons | Product reviews | Tech news | Dow

cnet NEWS.COM TECH NEWS FIRST

FRONT PAGE | ENTERPRISE SOFTWARE | ENTERPRISE HARDWARE | SECURITY | NETWORKING | PERSONAL TECH

SAVED STORIES 0

SEAR

The Net

# Eolas files motion to enjoin IE

Last modified: October 8, 2003, 11:29 AM PDT

By Paul Festa
Staff Writer, CNET News.com

PRINT EMAIL SAVE

**Eolas Technologies has filed a motion to permanently enjoin Microsoft's distribution of its Internet Explorer browser amid a flurry of court filings by both sides in the pivotal patent infringement case.**

Eolas, the sole licensee and sublicensor of a browser plug-in patent owned by the University of California, on Monday asked the U.S. District Court in Chicago for an injunction against distributing copies of IE capable of running plug-in applications in a way the Eolas patent covers.

"If they're not going to pony up and take a license under the patent, then they shouldn't be using it," Martin Lueck of Robins, Kaplan, Miller & Ciresi said in an interview.

**News.context**

**What's new:**
Patent holder Eolas files a motion to essentially stop Microsoft from distributing Internet Explorer, as both sides keep courthouse clerks hopping.

**Bottom line:**
Should the court grant the injunction, and find that Microsoft's proposed IE work-around does not circumvent the patent, the software giant may find itself forced to pay Eolas to provide fundamental plug-in capabilities in the browser.

HIGH IMPACT

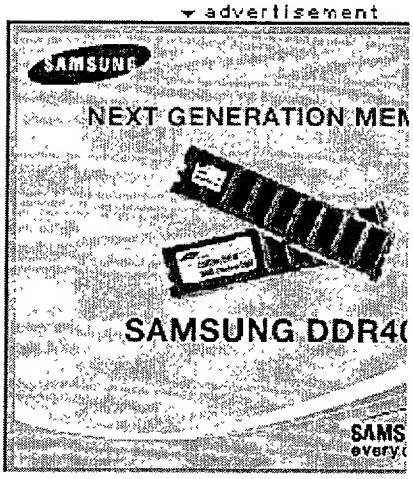
**For more info:**
More news on patents.

The Eolas patent infringement victory has rattled the Web since it was handed down in August. In its verdict, a jury found that Microsoft's IE browser infringed on an Eolas patent that describes how a browser opens external applications of the type Macromedia, Adobe Systems, RealNetworks, Apple Computer, Sun Microsystems and many other software providers produce.

Microsoft and the plug-in vendors aren't the only ones who are losing sleep over Eolas.

Web developers face the possibility of having to significantly rewrite their pages or strip them of commonly used technologies like Macromedia's Flash. And other browser makers, including Opera Software and two open-source development projects relied upon by

advertisement




GetUptoSpeed

Click for an in-depth look at:
- Enterprise security
- Open source
- Utility computing
- VoIP
- Web ser
- Wi-Fi

CNET DIGITAL L

**Digital home: multimedia den**
Your entertainment center and PC merge into a in this room in our digital home.

**Digital home: sensibly stylish living room**
Budget concerns don't mean that you can't trick living room in high style.

**Digital home: Lavish living room**
Movies and music have never looked or sounde than in this spare-no-expense retreat.

**Digital home: Functional family room**
From playing video games to making home vide functional family room pleases the whole brood.

**Digital home: Ultramodern living room**
If style is your game, the eye-catching gear in t will fit you like a glove.

**Top personal tech videos:**

Palm's new Tungst

companies like Hewlett-Packard and Apple, also face an uncertain future in terms of their plug-in technologies.

Lueck said Eolas would still permit Microsoft to distribute IE as is, as long as it's being used in conjunction with an application provider or a corporate intranet that has an Eolas plug-in license.

So far, Eolas has not granted any such licenses.

Lueck also noted that, should the motion be granted, Microsoft still could distribute IE with the plug-in capability disabled.

Microsoft said it is well on its way to side-stepping both the patent and a potential injunction with an IE alteration it previewed Monday--a version it expects to introduce early next year.

The previewed alteration would change the way IE renders pages that use ActiveX Controls to launch plug-ins. Microsoft also recommended to developers some methods of invoking external applications in a way it claims would circumvent the patented plug-in method.

Lueck and Eolas founder Mike Doyle said they were in the process of examining the IE preview and would not comment on its merits.

**Microsoft's appeal**

Microsoft on Monday filed motions to set aside the $521 million judgment and to grant it a new trial.

"As our court papers outlined (Monday), we believe we have substantial grounds for reconsideration by the judge," said Michael Wallent, a general manager in Microsoft's Windows division.

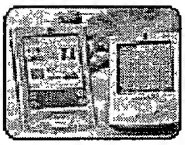
The Redmond, Wash., software giant asked for the new trial, citing several factors, including the unusual proportions of the jury's judgment and the court's refusal to allow discussion of some prior art or similar technology that Microsoft believes predated the Eolas patent and should therefore invalidate it.

Microsoft mentioned one piece of prior art in particular, the Viola browser, invented by Perry Pei-Yuan Wei, an artist, software engineer and then a student at the University of California at Berkeley. That browser dates back to 1991 and its plug-in capabilities to 1992, nearly two years before Eolas filed for its patent.

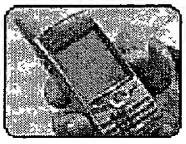
The motions Microsoft filed Monday are prerequisites for appealing the case to the U.S. Court of Appeals, which the company has pledged to do.

Microsoft also said it was preparing to challenge Eolas' demands for the court to update the damages award to include the period of September 2001 to the present. That could raise the amount of the award by hundreds of millions of dollars, though both sides declined to give a more exact calculation.

The $521 million award was calculated based on units Microsoft



mean business
ROLL VIDEO

Microsoft upgrades Windows XP Media
ROLL VIDEO


Treo 600 readies fo release
ROLL VIDEO

See more videos

- **This week's headlines**

**Latest headlines**

- Microsoft readies database add-on
- Linux inches up corporate IT priority
- IBM, Cisco push data center standa
- My (brief) career as an ISP
- Cable firms bet on broadband spee
- Sun to release Solaris for AMD's Op
- SEC filings: Revenue, profit...cybers
- Salon CEO resigns
- Feds nab suspected online conman
- Dell knows what do with dead PCs
- Schwarzenegger enlists Fiorina for
- Interwoven, iManage set date for m
- Palm looking into Tungsten glitch
- Handspring narrows its loss
- Corel trims staff after takeover

**Most popular headlin**

- Trillian connects with Yahoo yet age
- Microsoft's new security plan
- Nokia cites fake batteries in phone
- HP, Disney expand technology allia
- Court's call: Hands off VoIP
- Eolas files motion to enjoin IE
- Shift key breaks CD copy locks
- Apple to uncrate Panther OS this m
- Steam age tech takes heat off chips
- Yahoo keeps up profit streak

distributed between October 1998 and September 2001. Eolas has also calculated that Microsoft owes it about $111 million in interest on that award.

Even as both sides escalated the post-trial battle with the Monday filings and Microsoft's IE preview, Lueck called changes to IE unnecessary and reiterated that Eolas was willing to offer Microsoft a paid-up license in exchange for the standing jury verdict plus interest.

**Special report**
Patent politics ▸
A lawsuit has Microsoft rivals rushing to the defense of the software giant's Internet Explorer browser

"Eolas and the university are willing to resolve the case on a very reasonable basis," he said. "In view of the amount of the verdict and the accrued prejudgment interest, we'd be willing to give them a paid-up license, if they were willing to take out a license."

Lueck warned that the offer was not indefinite.

"That might change in the future, if they continue to refuse the deal," he said. "The quid pro quo would be settle it now--not force us to litigate for two, three, four years or whatever it is that they have in mind."

Microsoft contested Lueck's characterization of the offer as "reasonable" and said the company preferred to pursue its workaround strategy than sign a deal.

"In addition, the changes we rolled out for IE are modest and will not have significant impact on consumers or the Web community as a whole," Microsoft's Wallent said. "Based on that, the idea that we would pay more than $630 million to get rid of a single mouse click on a small fraction of Web pages is not something that we're entertaining."

Wallent based the "more than $630 million" figure on the $521 million verdict and a $111 million interest claim by Eolas.

## Related stories

- Eolas says it would settle over IE
  **September 19, 2003**
- IE patent endgame detailed
  **September 11, 2003**
- Microsoft ordered to pay $521 million
  **August 11, 2003**
- Microsoft steps up ActiveX security
  **July 21, 2003**
- **Get this story's "Big Picture"**

## Related quotes

QUOTES DELAYED 20+ MINUTES

- Microsoft Corporation **MSFT** 28.94 0.00 (0.00%)
- Apple Computer Inc **AAPL** 23.64 0.19 (0.81%)
- Adobe Systems Incorporated **ADBE** 41.87 -0.06 (-0.14%)
- RealNetworks Inc **RNWK** 7.31 0.05 (0.69%)

**White papers about Browsers** More results

- The Mobile Internet and Wireless Networking: Opportunities, Challenges and Solutions
  Axis Communications
- Web-To-Host: Your Future!
  ICOM Informatics

**CNET NEWSLETTERS**
CLICK ON A TITLE BELOW TO LEARN MORE:

- News.com Morning Dispatch sample
- News.com Afternoon Dispatch sample
- News.com Enterprise Hardware sample

All News.com newsletters

**SPECIAL OFFERS FROM OUR PARTNERS**
CLICK ON A TITLE BELOW TO LEARN MORE ABOUT IT.

- Business Management
- Small Business Owners
- IT Professionals

SIGN UP NOW

Manage My Newsletters



How to advertise | Support | CNET Jobs
Copyright 1995-2003 CNET Networks, Inc. All rights reserved.

Produced by Challenge

10/9/03 NYPOST 38
10/9/03 N.Y. Post 38
2003 WL 62825703
**(Publication page references are not available for this document.)**



New York Post
(c) 2003 N.Y.P. Holdings, Inc. All rights reserved.

Thursday, October 9, 2003

Business

# MICROSOFT RIVALS JOIN PATENT FIGHT ; M'SOFT RIVALS JOIN TO WAGE PATENT FIGHT

By STEPHEN LYNCH

Lawsuits make strange bedfellows. After Microsoft Corp. lost a patent suit in August and a Chicago jury ordered the software giant to fork over an astounding $521 million in past licensing fees, you would expect rivals like Sun Microsystems, Apple Computer and Macromedia, Inc. to be laughing up their sleeves.

Instead, the companies have banded together to help Microsoft appeal the case, fearful of what implications the patent decision could have on their own businesses - and the Internet itself.

"This affects almost every company that writes software for the Internet," said Janet Daly, a spokeswoman for the W3C, a consortium of Web developers.

The case centers around the embedding of small interactive programs in Web pages, called "plug-ins" or "applets."

The plaintiff, Michael Doyle, argues that such technology was developed in 1993 by his group at the University of California. A patent on the system, granted in 1998, is owned by the university and licensed to Doyle's company, Eolas Technology.

Microsoft uses the patented technology, which it calls ActiveX, in its Internet Explorer browser, Doyle argues. But Microsoft isn't the only company that uses this type of system. Apple, Macromedia, Sun and dozens of other firms also stream video or software through Web browsers - which could leave them open to similar challenges.

And the W3C, which sets Internet standards, has warned that the basic code underlying the Web, known as HTML, could infringe on the copyright as well.

When Microsoft lost to Doyle in court, software companies feared huge swaths of the Internet would need reprogramming - or owe Eolas licensing fees.

TABULAR OR GRAPHIC MATERIAL SET FORTH IN THIS DOCUMENT IS NOT DISPLAYABLE

Copr. © West 2003 No Claim to Orig. U.S. Govt. Works

WITHOUT A LICENSE: Microsoft Corp., led by Bill Gates (left) and Steve Ballmer, has been ordered to pay $521 million in a patent case that has rivals fretting.AP

---- INDEX REFERENCES ----

COMPANY: Apple Computer Inc (APPLC); Macromedia Inc (MACROM); Microsoft Corp (MCROST); Sun Microsystems Inc (SUNMIC)

NEWS SUBJECT: (Patents (C133); Regulation/Government Policy (C13); Corporate/Industrial News (CCAT); Intellectual Property (CGYMTR); Industrial Property (CINPRP))

INDUSTRY: (Computers/Electronics (I3302); Software (I330202); Systems Software (I3302020); Applications Software (I3302021); Computer Programming (I83941); Internet Browsers (IBROWS); Desktop Computers (I3302003); Computing (ICOMP); Computer Hardware (ICPH); Internet/Online Services (IINT))

REGION: (United States (USA); North American Countries (NAMZ))

Word Count: 310

10/9/03 NYPOST 38

END OF DOCUMENT

Copr. © West 2003 No Claim to Orig. U.S. Govt. Works

hU2T20" Smh2E80

# Ray Ozzie's Weblog

home

stories

## Saving the Browser

Some months back I became aware of the patent US 5,838,906 and the Eolas lawsuit against Microsoft, and followed a bit of conversation on the Net related to it. As many, I believed the issue would quickly go away because of ample prior art. Regrettably, this seems not to be the case.

It now seems that perhaps the browser itself and the browsing experience may have to be nontrivially modified as a result of the judgment. Although a bit late, if some of us perhaps dust off our old code, is there a chance that we could still save the browser through demonstration of clear prior art?

For my own interest, and for the record, I recently spent a little time pursuing my intuition that Lotus Notes R3 might be viable prior art relative to the patent in question. I am not an attorney, and I am surely not well versed in the nuances of the case, but it seems to me after initial investigation that there is indeed quite a bit of relevance.

I pursued this with the assistance of my brother, Jack Ozzie, and with another of my employees at Groove, Rob Slapikoff. Both Jack and Rob worked for me at Iris Associates in the development of Lotus Notes. Although I am personally responsible for a good deal of the "browser" code in question, I asked Jack to help because he specifically did all of the work related to our "object linking and embedding" technologies - first a Lotus technology referred to internally as DIP, (Document Interchange Protocol), and later in loose collaboration with the Windows & Excel teams on what was referred to as either CDP (Compound Document Protocol) or OLE (Object Linking and Embedding). I asked Rob to help because he was, in essence, a Lotus Notes "solution developer" at the time, and was very familiar with how one would quickly weave together a solution involving multiple applications.

When I began this investigation, I thought that it might be challenging to recreate a scenario, given the feature set available Notes R3, that was close to what was described in the patent. In fact, however, the hard part was only in putting together a computing environment that ran Notes R3. Once we had Notes running, it only took about 15 minutes to reproduce what I've shown below, and there was no programming involved. Meaning, everything done was done with just the out-of-the-box UI of both Notes and Excel.

First, let me describe the environment that we recreated. Since the filing date on the patent is October 17, 1994, I sought to obtain software that was clearly shipping to end-users before that date. I set about to assemble the following software to assist in the demonstration: Microsoft MS-DOS 6.22, Microsoft Windows for Workgroups 3.11, Microsoft Excel 5.0, and Lotus Notes 3.0. In my personal archives, I happened to be in possession of DOS, Windows, and a freshly shrink-wrapped copy of Notes. I selected Microsoft Excel 5 because information on the Web indicated that it shipped 12/14/93, and I easily obtained a shrink-wrapped copy via eBay in a matter of days.

I first used VMware Workstation 4 to create a virtual machine environment roughly comparable to that of the era. Then, I installed MS-DOS 6.22 within that virtual machine, as well as Windows for Workgroups 3.11. Finally, I installed Excel 5.0 and Notes 3.0. I chose WFW because I felt it to be very important to create a configuration that could be used as a "client/server"

network environment between multiple virtual machines. As such, I installed both the Notes 3.0 client and server programs, and set about to creating the demonstration herein.

For those of you not familiar with Lotus Notes, but let me give you a very brief overview. I founded a company, Iris Associates, in December of 1984 to create the product eventually purchased and marked by Lotus Development Corporation as Lotus Notes. Notes first shipped in December of 1989, and at the time of this writing is in use by more than 100 million individuals at major corporations worldwide.

The product was and is an inherently "networked" client/server product. It consists of a client piece that is used to browse and create text, hypertext, and hypermedia documents, and collections of such that are woven into "applications" and "solutions". These documents, applications and solutions are *hosted* on a server analogous to today's "Web application servers". Notes documents are, in essence, compound documents analogous to today's "HTML documents". They structurally consist of a stream of "CD Records" (Compound Document Records) which are, in essence, the logical equivalent in an HTML document of HTML "elements" or tags. The very name "Compound Document" chosen for such CD Records was chosen in the late 80's during Notes development, and is indicative of how we envisioned documents as being compound combinations of different media elements. The format of CD Records was documented in the Notes developer toolkits.

The Lotus Notes client, which herein I also refer to as the "browser", could use a number of methods to fetch documents from (or store documents to) the Notes Server. One way was through "document links", which are analogous to the Web's URLs, being "pointers" to documents. The general process used by the Notes browser to fetch and display a document was quite similar to that used by a Web browser: the Notes client would parse the link to figure out what server it needed to fetch the document from, it would then send a network transaction to that server, then the server would read the document from its internal storage, and respond back to the client with the contents of the document. Then, the Notes browser would begin sequentially parsing the series of records contained within, rendering the contents on-screen in a window as the records were being parsed.

Through use of "document links" that are used to create webs of interlinked Notes documents, Notes is naturally a hypertext system. In fact, in creating Lotus Notes I was quite influenced by the writings of Ted Nelson, about 10 years prior to starting work on it. In reading the patent, one of the more interesting questions pertained to embedded links to hypermedia documents. In fact, one of the "big deals" about the release of Lotus Notes R3 was its support of embedded media, and the inclusion of linking technology to enable networked hypermedia. To drive this point home, note that two diskettes were included in the Lotus Notes package, shrink wrapped along with the Lotus Notes diskettes themselves. One is entitled "Lotus Multimedia Tools" and the other is "Lotus Media Clips" with sample content.

```
Directory of Lotus Multimedia Tools
[.]             [..]            ANGELSNG.WA_    ANNDLL.DL_      ANNICON8.DL_
ANNICONE.DL_    ANNICONV.DL_    ANNOTATE.EX_    ANNOTATE.HL_    ANNOTATE.SMI
COMMDLG.DL_     INSTALL.EXE     LMPDLL.DL_      LMPHNDLR.DL_    LOTUSSND.EX_
LOTUSSND.HL_    LSM.DL_         LTSANNOT.BM_    LTSANNOT.MAC    LTSANNOT.SM_
LTSICN02.DL_    LTSMEDIA.BM_    LTSMEDIA.MAC    LTSMEDIA.SM_    LTSSOUND.BM_
LTSSOUND.MAC    LTSSOUND.SM_    LZEXPAND.DLL    MEDMAN.EX_      MEDMAN.HL_
OLECLI.DL_      OLESVR.DL_      SHELL.DL_       VER.DLL         WINHELP.EX_
WINHELP.HL_
              34 File(s)      1,079,263 bytes

Directory of Lotus Media Clips
[.]             [..]            BASSOON.WA_     BELL.WA_        CHARGE.WA_
```

```
COOLJAZZ.WA_    DEATHMAR.WAV    DRUMROLL.WA_    FANFARE1.WA_    FANFARE2.WA_
FANFARE3.WA_    FANFARE4.WA_    FLUTEDUO.WA_    HOORAY.LS_      INSTALL.EXE
JAZZY.WA_       LZEXPAND.DLL    MELODRAM.WA_    OFF2RACE.WA_    PIZZICAT.WA_
RASPBERY.WA_    ROCKGUI.WAV     SILLYSNG.WA_    STEELDRM.WA_    SUSPCHRD.WA_
VER.DLL         WHEELO.LS_      WHOOSH.WA_
                26 File(s)      1,158,326 bytes
```

These were programs written by separate groups at Lotus that were "bundled" with the Lotus Notes application. As you can see by the content of these diskettes, Lotus' concept was that documents could be "annotated" in a live fashion with linked-to media.

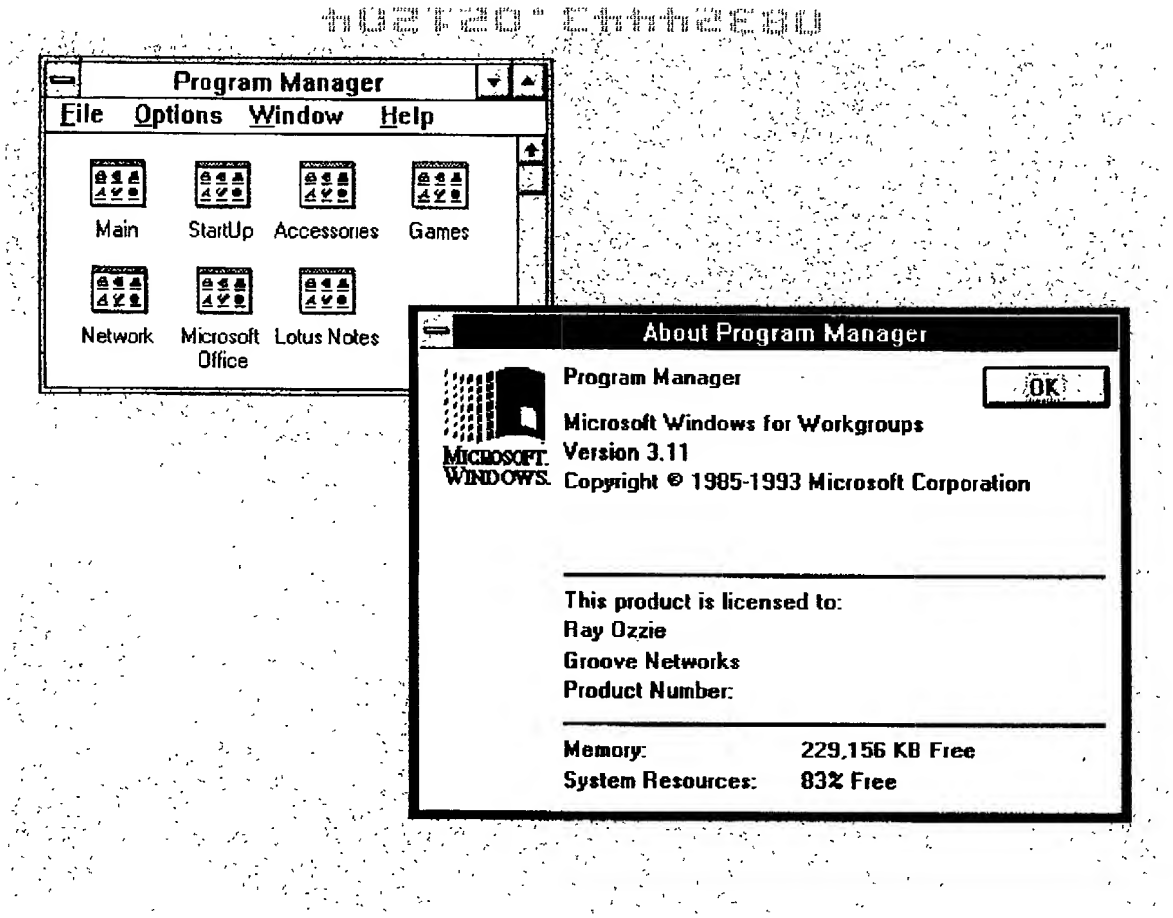
Note also that these programs used the Microsoft OLE libraries as their linking & embedding technologies, like Microsoft Excel itself. This is because at the time this was the easiest way to write programs using Microsoft OLE.

That said, Lotus Notes itself did not utilize the Microsoft OLE libraries, because the DDE-based linking and embedding support in Lotus Notes was developed prior to the emergence of OLE (as Lotus' own DIP protocol). When OLE came along, developers at my company worked with developers at Microsoft (in particular, Ed Fries of the Excel team) to ensure the compatibility and interoperability of Lotus Notes and such OLE-embedded programs, via standardization on DDE-based inter-application protocols. When OLE was first publicly introduced to the world, we were on stage with Microsoft demonstrating such interoperability.

In 1993 or thereabouts, we saw the emergence of TCP/IP, HTML, HTTP, Mosaic and the Web. From our perspective, all of these were simplistic emulations of a tiny subset of what we'd been doing in Notes for years. TCP/IP instead of Netbeui or IPX/SPX, HTML instead of CD records, HTTP instead of the Notes client/server protocols, httpd instead of a Notes server. And we were many years ahead in other ways: embedded compound objects, security, composition of documents as opposed to just "browsing" them, and a sophisticated development environment. I am quite embarassed to say that we frankly didn't "get" what was so innovative about this newfangled "Web" thing, given the capabilities of what had already been built.

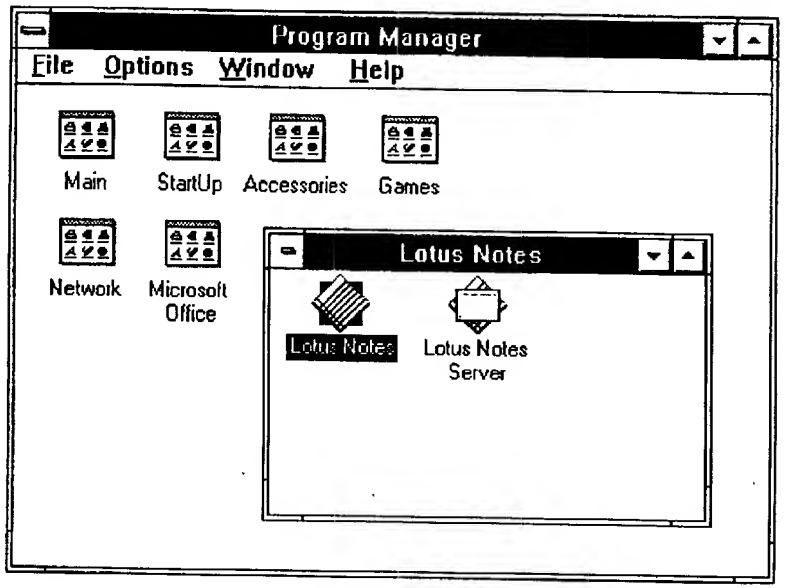
The following series of screen snapshots (of the VMWare emulator window) walks you through a simple demonstration of Notes R3's capabilities.

01 Windows Splash Screen



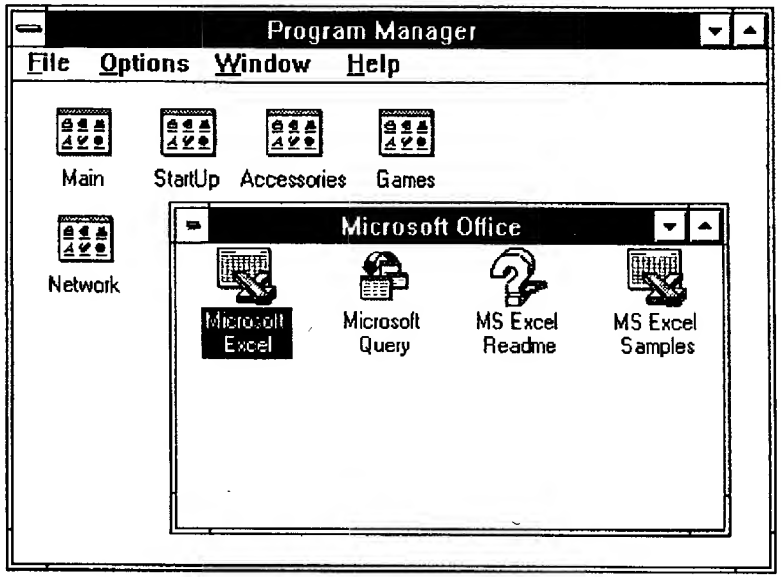

This just shows the general operating environment of this demo, and the rough timeframe in which Windows for Workgroups was prevalent.

02 Notes Program Group



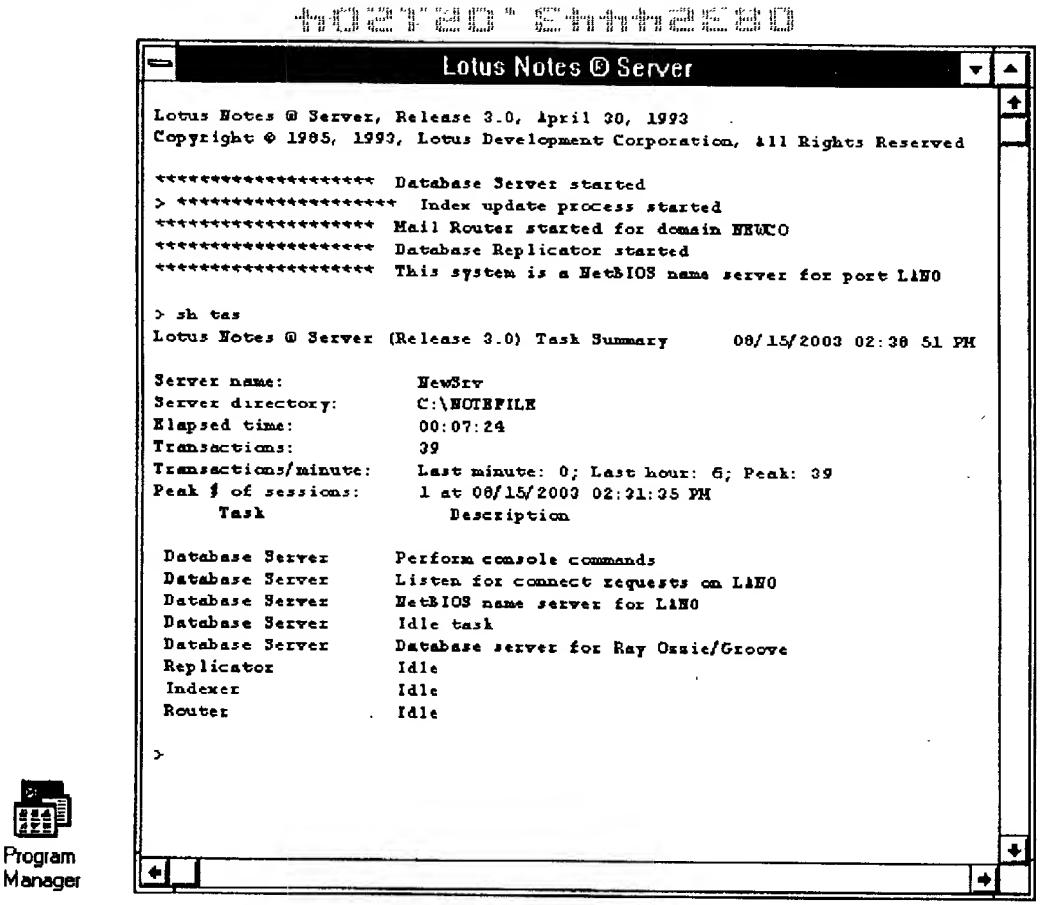

This is to show what particular components I installed. This is the Lotus Notes for Windows Release 3. Note that the Lotus Notes client and server programs were also available for OS/2 in Release 3.0. Some months later, in a later maintenance release of Release 3, we also shipped the product for Windows NT (client and server), for the Apple Macintosh (client only), and for various flavors of Unix.

03 Office Program Group



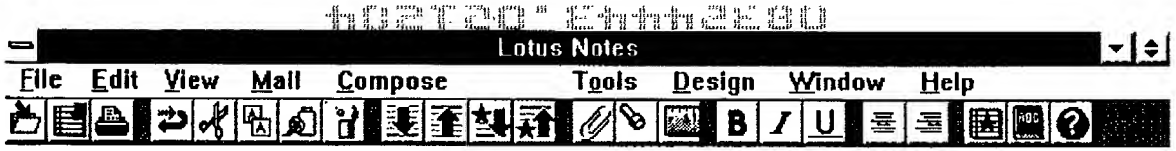

This shows what components of Excel 5.0 were installed.

04 Notes Server Console



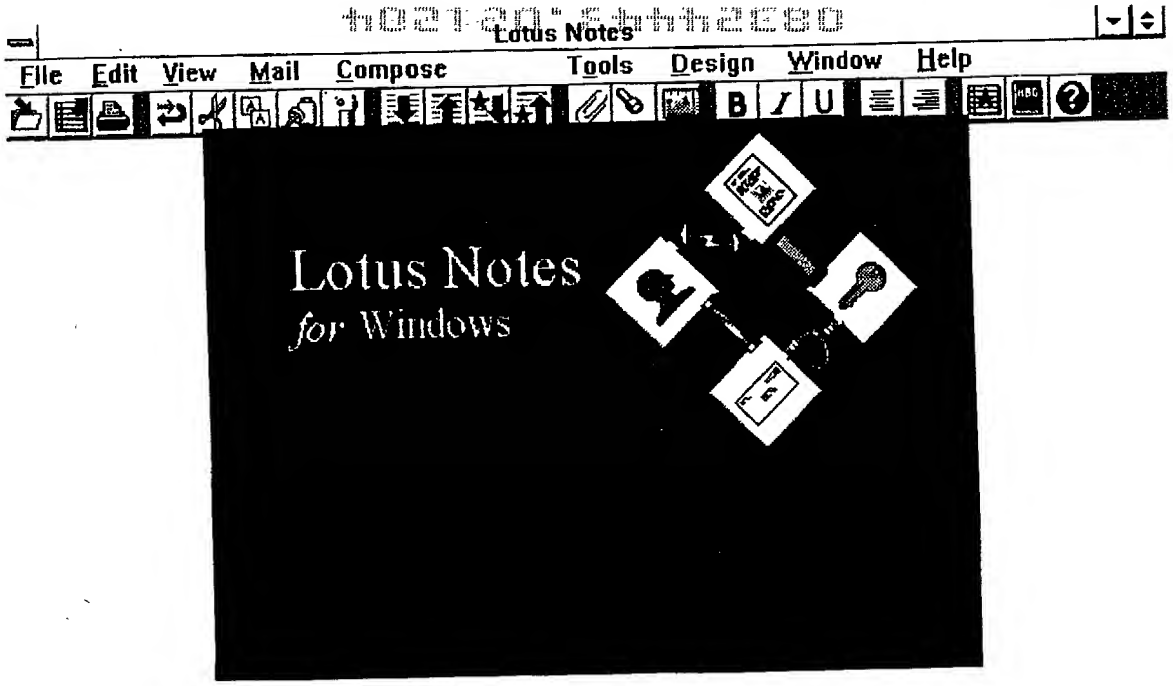

This shows the Lotus Notes R3 for Windows Server console while it is running. It is clearly using NETBIOS as the network transport protocol. This shows one remote user (Ray Ozzie/Groove) connected into the server in order to perform network transactions such as fetching documents.

05 Notes Browser Window



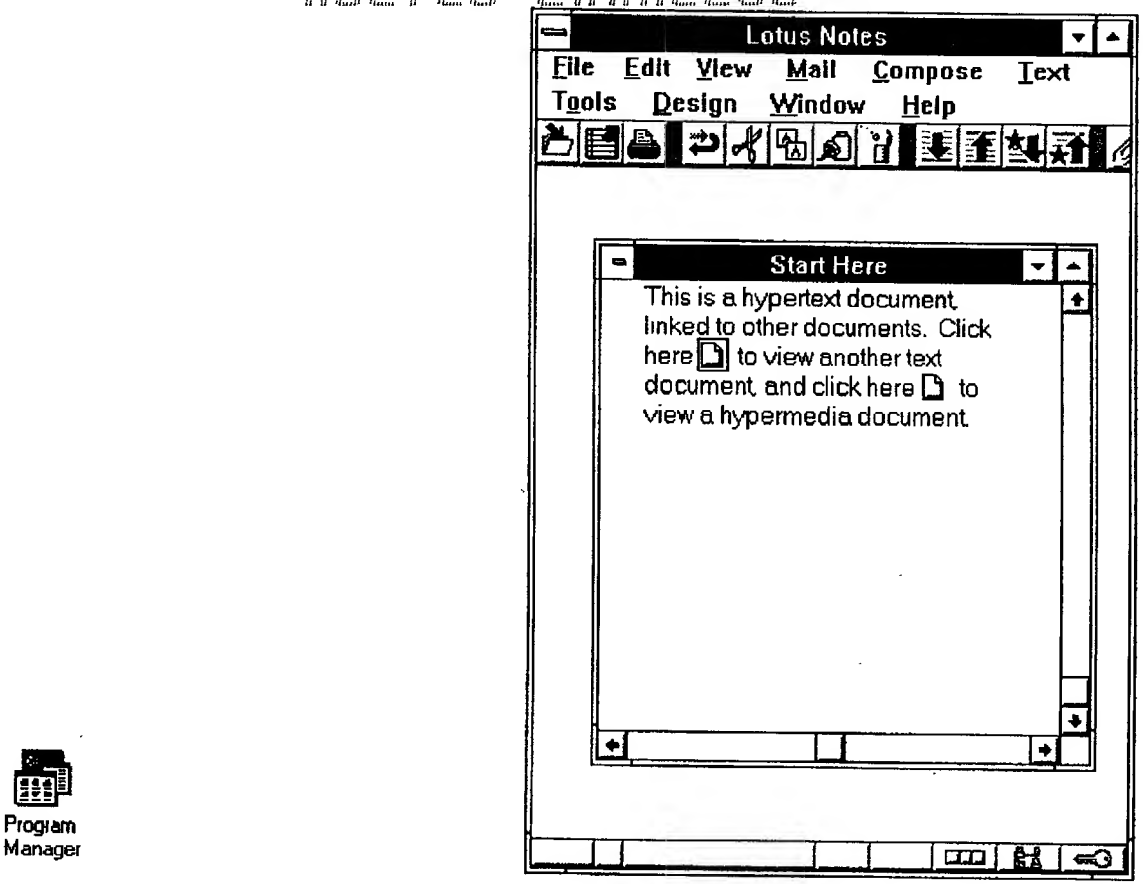
This is the Lotus Notes client user interface, maximized to full-screen. The "Workspace" icon in the center, when maximized, is a container where the user can remember "bookmarks" to commonly-used documents and applications. Applications in Notes are referred to as "databases", and are analogous to Web Sites in that they are collections of documents and application logic brought together for a specific purpose.

06 Notes Splash Screen




This is the Help About display, showing the precise version of Lotus Notes in use for this demonstration.

07 About to click Hypertext Link

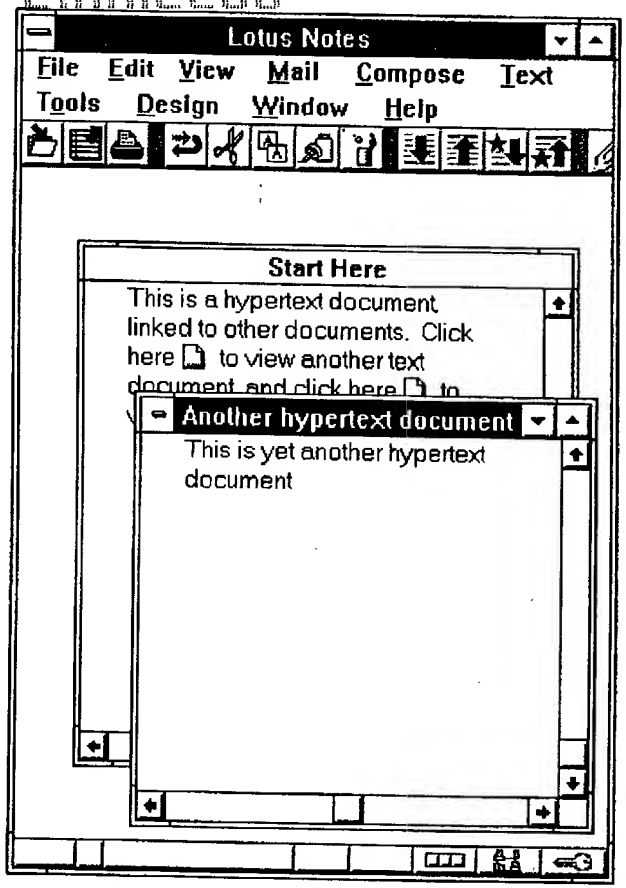



In this example, I have first resized the Notes browser window to only be a part of the screen, for reasons you'll see later.

Then, I used a Notes Workspace "bookmark" to fetch and open a document called "Start Here", which is a Notes document (consisting of CD records) fetched over the network from the Notes Server. Such a document is analogous to a Web Page, and it's being browsed here in the Lotus Notes client. This document has some text, and has two inter-document "links" that could potentially be pointing to other documents on this server, or on other servers.

Now, I'm going to double-click the first link (the one that's highlighted), which will open the linked-to document into a new browser window...

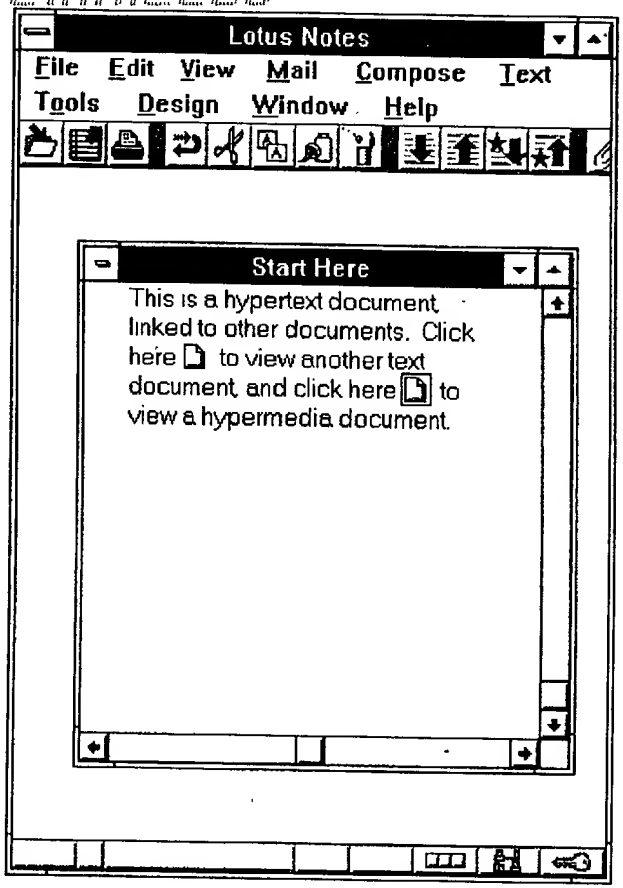
08 After clicking Hypertext Link




This is what things look like immediately after I double-clicked that first link. The Notes browser opened another window on the screen, and the linked-to document was fetched, and displayed as it was parsed.

Now, I close this window and return to the original document...

09 About to click Hypermedia Link

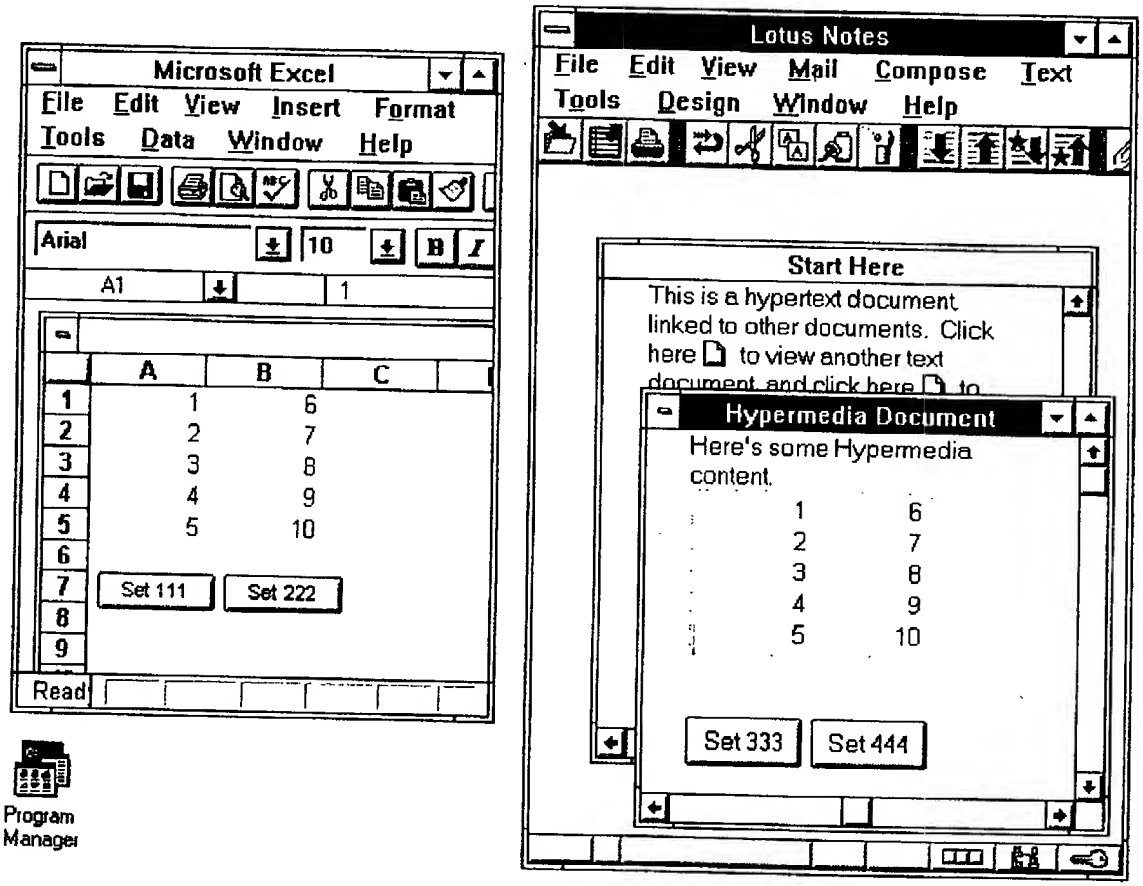



Now, as you can see, I immediately double-click this second link, which again opens a new document into a new window. But in this case, the linked-to document is a document with an embedded hypermedia object reference, and you're about to see what happens.

In essence, the linked-to document we're about to open has embedded CD records that describe 1) that an object is to be embedded into the page, 2) the class of object (in this case "EXCEL"), 3) the pointer to the object (in this case X:TEST.XLS", which is a reference to a spreadsheet object that is potentially on a remote server computer represented by drive letter X., and 4) a parameter indicating whether or not this object should be immediately activated upon browsing, or whether the object needs to be double-clicked before activation.

In this example, we've set up the object reference on the linked-to page so that it does automatic activation, so quite a bit will be happening when we just simply double-click the link in the above document. Here goes...

10 After clicking Hypermedia Link



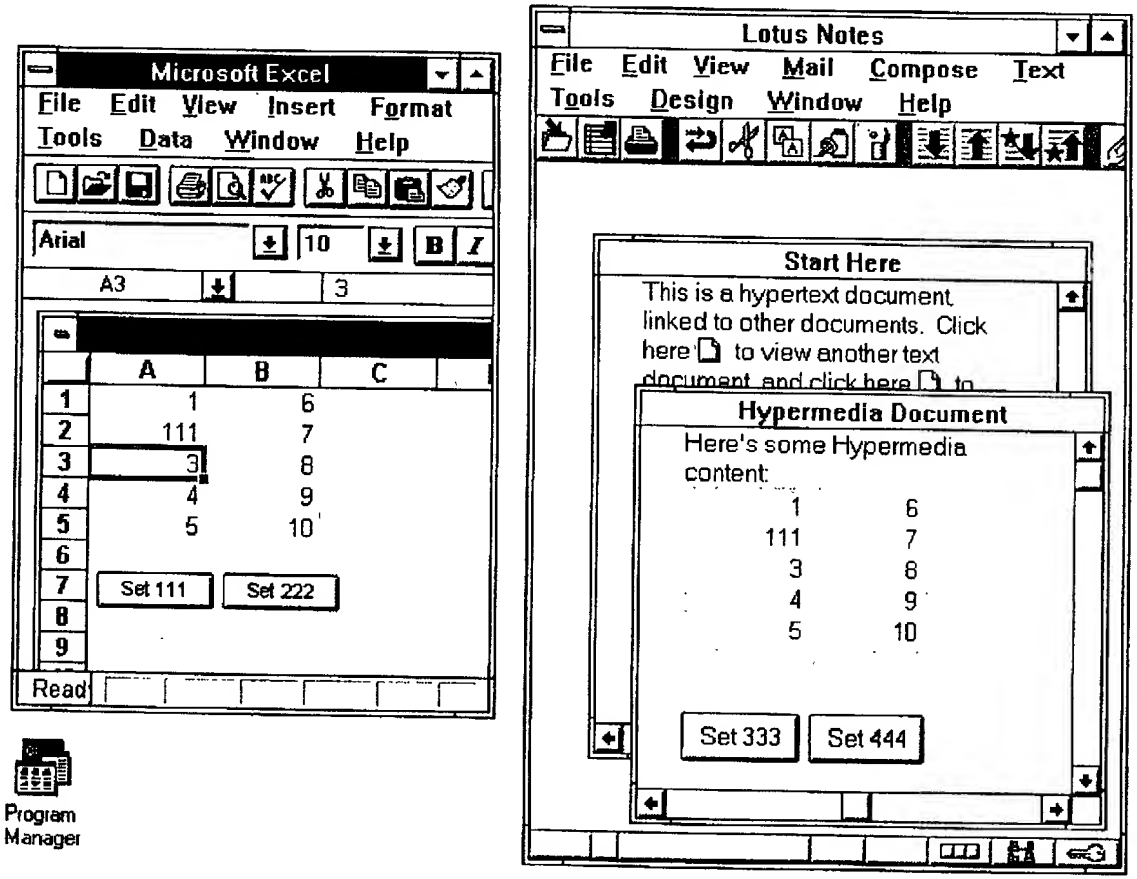

Everything that appears on the screen above appeared automatically as a result of simply opening the linked-to hypermedia document. Meaning, when I took the screen shapshot above, I hadn't clicked or typed anything beyond the double-click to follow the link.

If you look at the "Hypermedia Document" window, you'll first see some text that was rendered, and then you'll see a rectangle displaying the spreadsheet, then you'll see two buttons. The rectangle is an up-to-date rendering of the "live" current state of an external object, as you'll see in a moment. Note that as the internal CD object tag was parsed by the Notes browser, the external Excel application was launched, and the referenced spreadsheet was activated into it. Then, an active inter-connection between the Notes browser client and the external application was initiated, and the rendering of the object was sent from the external application to the browser client. This inter-application connection remains intact. All communication between applications is done using protocols riding on the DDE Windows messaging transport.

To demonstrate the "live" nature of the interconnection, I've previously placed two buttons on the spreadsheet itself, which

assign the value of cell B2 to a fixed, known value. Similarly, I've previously placed two buttons within the Hypermedia Document that communicate with the active, linked-to application, telling it also to set the value of cell B2 to a fixed known value.

11 After clicking Set 111 button in Excel

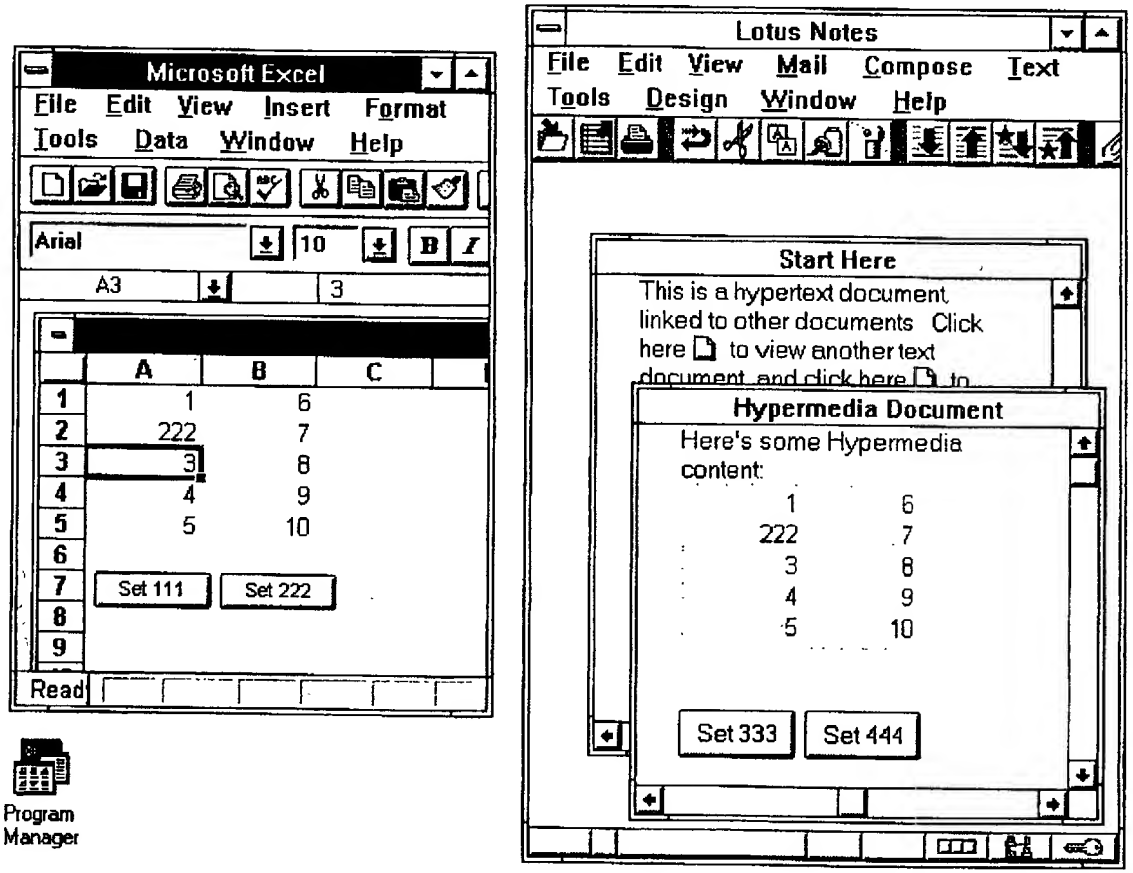



So, as I click the "Set 111" button, note that it sets the cell B2 to 111. Note that not only is the spreadsheet updated, but also the active rendering of that spreadsheet object is updated in the Hypermedia Document. Thus, you can see that the two applications are actively inter-connected.

As you watch this demonstration, it's important to note that this version of Excel was the first to have ODBC support. Thus, the cells being displayed, or perhaps a chart, could be the result of a remote query to a very large SQL database. Furthermore, in this example, setting the value of a cell could just as well have been the "trigger" to a series of very complex commands within the spreadsheet itself – meaning that the application could have been far more robust than just setting the

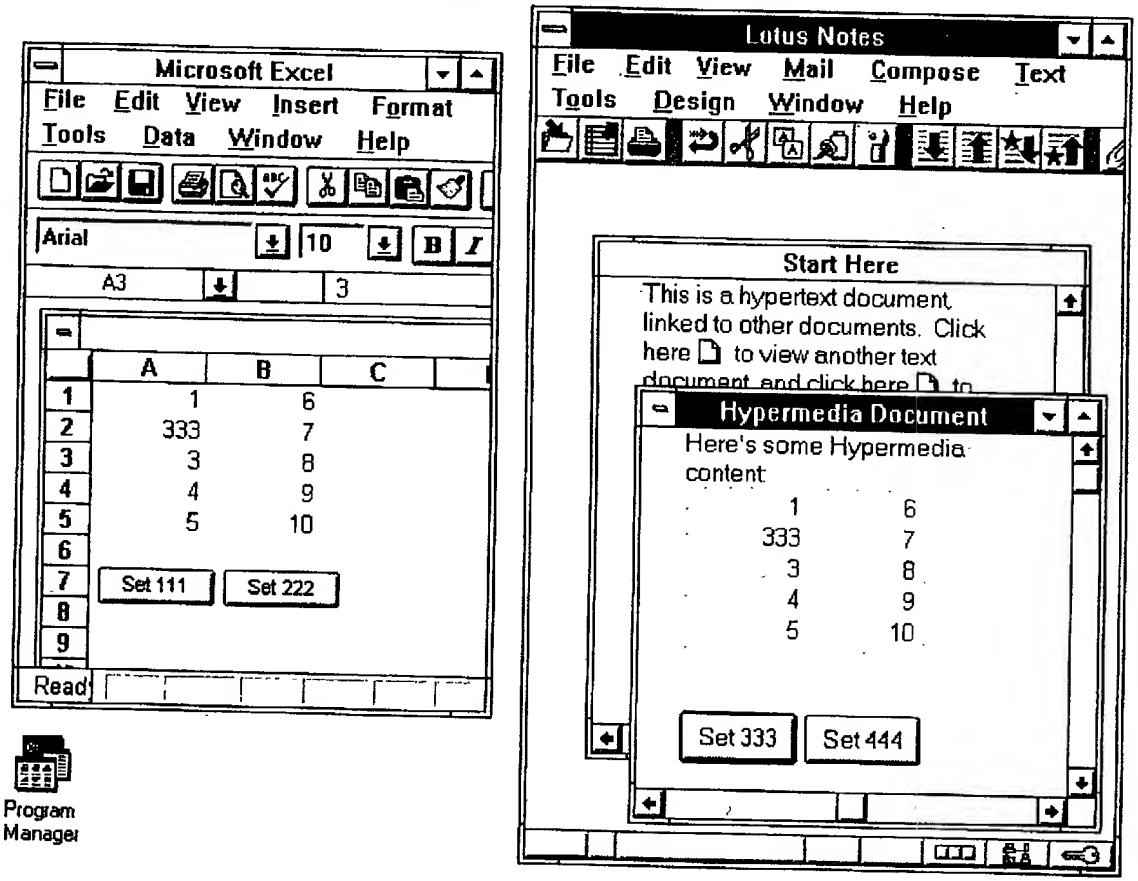
value of a cell.

12 After clicking Set 222 button in Excel




Again, just to prove the point that you could have multiple "controls" in the spreadsheet viewer itself, the second button sets the value to "222".

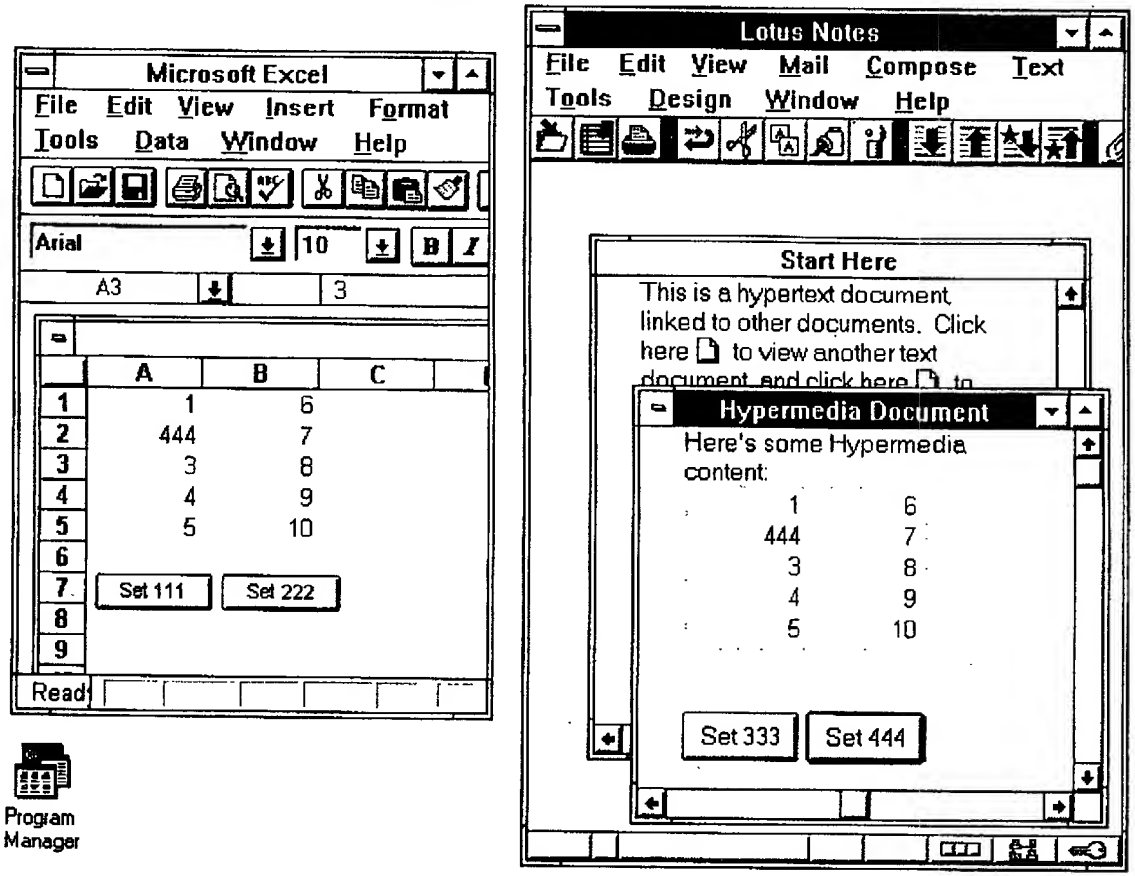
13 After clicking Set 333 button in Browser




Now, for something completely different, I click the button within the Hypermedia Document called "Set 333". This actually uses the active inter-application connection via DDE to, in essence, "drive" the separate application. In this case, it's setting the value of that same cell to 333. Again, remember that this might've instead caused a fetch of a small piece of data from a very large remote data set.

And again, note that not only is the spreadsheet updated, but also the active rendering of that spreadsheet object is updated in the Hypermedia Document. Thus, you can see that the two applications are actively inter-connected.

14 After clicking Set 444 button in Browser.




And finally, I just set it to 444 to show that you can have multiple commands in the browser document that do different things

Now, let me take a very brief look at the claims in the patent, and quickly review how I believe this relates to this particular prior art.

"A0220" Ethernet

What is claimed is:

1. A method for running an application program in a computer network environment, comprising:

providing at least one client workstation and one network server coupled to said network environment, wherein said network environment is a distributed hypermedia environment;

executing, at said client workstation, a browser application, that parses a first distributed hypermedia document to identify text formats included in said distributed hypermedia document and for responding to predetermined text formats to initiate processing specified by said text formats; utilizing said browser to display, on said client workstation, at least a portion of a first hypermedia document received over said network from said server, wherein the portion of said first hypermedia document is displayed within a first browser-controlled window on said client workstation, wherein said first distributed hypermedia document includes an embed text format, located at a first location in said first distributed hypermedia document, that specifies the location of at least a portion of an object external to the first distributed hypermedia document, wherein said object has type information associated with it utilized by said browser to identify and locate an executable application external to the first distributed hypermedia document, and wherein said embed text format is parsed by said browser to automatically invoke said executable application to execute on said client workstation in order to display said object and enable interactive processing of said object within a display area created at said first location within the portion of said first distributed hypermedia document being displayed in said first browser-controlled window.

Notes has a client workstation and a network server, coupled by the network, and this is the distributed hypermedia environment. The Notes client is the browser application that parses the distributed hypermedia Notes document to identify text formats, setting up links and buttons, etc. When opened, at least a portion of a hypermedia document is displayed on the client, which has contained within it a compound document record indicating that an object is embedded. The object reference has type information associated with it that enables it to identify the executable application, and the browser can automatically invoke the application to execute on the client to display and enable interaction with the object. The display area within the

document is processed interactively, dynamically and interactively updating when controls are manipulated either within the browser application or within the executable application itself.

2. The method of claim 1, wherein said executable application is a controllable application and further comprising the step of:

interactively controlling said controllable application on said client workstation via inter-process communications between said browser and said controllable application.

Indeed, the executable application is controllable and can be controlled via DDE-based inter-process communication between the Notes browser and the external application.

3. The method of claim 2, wherein the communications to interactively control said controllable application continue to be exchanged between the controllable application and the browser even after the controllable application program has been launched.

Again, for the entire duration that the browser window is being displayed, there is active communications between the browser application and the controllable external application.

4. The method of claim 3, wherein additional instructions for controlling said controllable application reside on said network server, wherein said step of interactively controlling said controllable application includes the following substeps:

issuing, from the client workstation, one or more commands to the network server;

executing, on the network server, one or more instructions in response to said commands;

sending information from said network server to said client workstation in response to said executed instructions; and processing said information at the client workstation to interactively control said controllable application.

5. The method of claim 4, wherein said additional instructions for controlling said controllable application reside on said client workstation.

I didn't demonstrate this, but imagine simply that we had enhanced the demonstration such that the Excel spreadsheet had used ODBC to issue queries to a very large data set on the network server running a SQL server. In this case, one could press a button on the client workstation (in either the browser or Excel), the query would be execute on the network server, and would send the result set back. The client application would then respond to the data and would have functions or macros that would behave differently based upon the data returned from the server. Thus, the client workstation would be using the retrieved information to further control the client application.

Finally, claims 6-10 are identical to claims 1-5, with the substitution of "The computer program product" instead of "The method". Well, yes, we did create an actual *product* to do such a thing. We even shipped it about 18 months before his filing. Lotus was a public company and at the time one of the biggest forces in the personal computing industry, so surely the person or persons doing the patent filing must have or should have known about our hypermedia innovations. Given all the press coverage, he was likely also influenced by them in envisioning his own distributed hypermedia enhancements to the then-nascent Web browser technology.

##

© Copyright 2003 Ray Ozzie.
*Last update: 9/13/2003; 10:12:27 PM.*

h021120" Ehth2E80



## Controversial Patents

Click on the patent number in the Lookup column to examine the patent in IBM's Patent Database.

| Title | Holder | Date Issued | Summary | Lookup |
|---|---|---|---|---|
| **One-Click Ordering** | Amazon | 09/28/1999 | Method and system for placing a purchase order via a communications network | 5960411 |
| **Indexing the Web** | Digital/CMGI | January 26, 1999 | This patent, "a method for parsing, indexing and searching world wide web pages," is the Altavista patent for web crawlers, spiders and robots. | 5864863 |
| **Affiliate Program** | Amazon | 02/22/2000 | Internet-based referral system that enables individuals and other business entities ('associates') to market products, in return for a commission, that are sold from a merchant's website | 6029141 |
| **XPointer** | Sun Microsystems | Aug. 19, 1997 | This "method and system for implementing hypertext scroll attributes" was obtained by Jakob Nielsen and Sun Microsystems. Sun claims that this patent affects the W3C's Xpointer activity. | 5659729 |

| | | | | |
|---|---|---|---|---|
| **Style Sheets** | Microsoft | 01/12/1999 | The use of style sheets in an electronic publishing system | 5860073 |
| **P3P** | Intermind | 01/19/1999 | Computer-based communication system and method using metadata defining a control structure. (P3P is "Platform for Privacy Preferences") | 5862325 |
| **WAP** | GeoWorks | 07/05/1994 | A flexible user interface for mobile communications devices | 5327529 |
| **Web-page Downloading** | Sony | 11/02/1999 | Apparatus for and method of automatically downloading and storing internet web pages. | 5978807 |
| **Embedded Hypermedia** | Michael Doyle, UC Regents | 11/17/1998 | Distributed hypermedia method for automatically invoking external application providing interaction and display of embedded objects within a hypermedia document. | 5838906 |
| **Error Handling** | MCI | 10/26/1999 | Hierarchical error reporting system | 5974568 |
| **Web Advertising** | Double Click, Inc. | 09/07/99 | Method of Delivery, Targeting, and Measuring Advertising over Networks | 5948061 |
| **GIF** | Unisys | 12/10/1985 | High speed data compression and decompression apparatus and method | 4558302 |
| **Selling Airline Tickets** | Priceline | 04/27/1999 | Method and apparatus for the sale of airline-specified flight tickets | 5897620 |
| **Web User Tracking Across Sites** | Infonautics | 01/27/1998 | Method and apparatus for attaching navigational history information to universal resource locator links on a world wide web page | 5712979 |
| **E-commerce Tracking** | Stephen Dale Messer | 11/23/1999 | Data processing system for integrated tracking and | 5991740 |

[First Name]

[Last Name]

[E-mail Address]

[Company Name/I

[Phone Number]

[Zip/Postal Code]

[Country]

You agree that this information is passed on to Thawte and that you will be contacted via e-mail and/or telephone by a member of the Thawte sales team. Your information will NOT be made available to anyone else.

submit

| | | | | |
|---|---|---|---|---|
| | | | management of commerce related activities on a public access network | |
| **Display of Spherical Images (VR)** | Interactive Pictures Corporation | 11/23/1999 | Method and apparatus for the interactive display of any portion of a spherical image | 5990941 |
| **Stateless Shopping Cart** | Sun Microsystems | 04/28/1998 | Stateless shopping cart for the Web | 5745681 |
| **E-commerce Sales** | Open Market | 02/03/1998 | Network sales system | 5715314 |
| **One-Button Sync** | 3Com | 12/07/1999 | Extendible method and apparatus for synchronizing multiple files on two different computer systems | 6000000 |
| **Y2K Fixes** | Bruce Dickens, McDonnell Douglas | 10/08/1998 | Date formatting and sorting for dates spanning the turn of the century | 5806063 |

If you know of controversial patents that we should add to our list, send them to Dale Dougherty.



Copyright © 2000-2003 O'Reilly & Associates, Inc. All Rights Reserved.
All trademarks and registered trademarks appearing on the O'Reilly Network are the property of their respective owners.
For problems or assistance with this site, email help@oreillynet.com